Parallel Metaheuristic Search

28

Teodor Gabriel Crainic

Contents

Abstract

The chapter presents a general picture of parallel meta-heuristic search for optimization. It recalls the main concepts and strategies in designing parallel meta-heuristics, pointing to a number of contributions that instantiated them for neighborhood- and population-based meta-heuristics, and identifies trends and promising research directions. The focus is on cooperation-based strategies, which display remarkable performances, in particular strategies based on asyn-

T. G. Crainic (✉)
CIRRELT – Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Montréal, QC, Canada

School of Management, Université du Québec à Montréal, Montréal, QC, Canada
e-mail: TeodorGabriel.Crainic@cirrelt.net

R. Martí et al. (eds.), *Handbook of Heuristics*,
https://doi.org/10.1007/978-3-319-07124-4_40

chronous exchanges and the creation of new information out of exchanged data to enhance the global guidance of the search.

Keywords
Parallel computation · Parallel meta-heuristics · Functional and data decomposition · Independent multi-search · Synchronous and asynchronous cooperative search · Integrative cooperative search

Introduction

Meta-heuristics often offer the only practical approach to addressing complex problems of realistic dimensions and are thus widely acknowledged as essential tools in numerous and diverse fields. Yet, even meta-heuristics may reach quite rapidly the limits of what may be addressed in acceptable computing times for many research and practice problem settings. Moreover, heuristics do not generally guarantee optimality, performance often depending on the particular problem setting and instance characteristics. Robustness is therefore a major objective in meta-heuristic design, in the sense of offering a consistently high level of performance over a wide variety of problem settings and instance characteristics.

Parallel meta-heuristics aim to address both issues. Their first goal is to solve larger problem instances than what is achievable by sequential methods and to do it in reasonable computing times. In appropriate settings, such as cooperative multi-search strategies, parallel meta-heuristics also prove to be much more robust than sequential versions in dealing with differences in problem types and characteristics. They also require less extensive, and expensive, parameter-calibration efforts.

The objective of this chapter is to paint a general picture of the parallel optimization meta-heuristic field. Following [40], it recalls the main concepts and strategies in designing parallel meta-heuristics, pointing to a number of contributions that instantiated them for neighborhood- and population-based meta-heuristics, and identifies a number of trends and promising research directions. It focuses on cooperation-based strategies, which display remarkable performances, reviewing in somewhat more depth the recently introduced integrative cooperative search [96]. Notice that parallel meta-heuristic strategies are examined and discussed from the conceptual, algorithmic design point of view, independent of implementation on particular computer architectures.

The parallel meta-heuristic field is very broad, while the space available for this chapter is limited. In addition to the references provided in the following sections, the reader may consult a number of surveys, taxonomies, and syntheses of parallel meta-heuristics, some addressing methods based on particular methodologies, while others address the field in more comprehensive terms. Methodology-dedicated syntheses may be found in [5, 84–86, 133] for parallel simulated annealing, [20, 21, 107, 119, 147] for genetic-based evolutionary methods, [32, 43, 45, 82, 170] for tabu search, [73] for scatter search, [18, 62, 92] for ant colony methods, and [116]

for variable neighborhood search (VNS). Surveys and syntheses that address more than one methodology may be found in [2, 4, 33, 36–38, 40, 50, 51, 90, 97, 125, 168].

The chapter is organized as follows. Section "Meta-heuristics and Parallelism" is dedicated to a general discussion of the potential for parallel computing in meta-heuristics, a brief description of performance indicators for parallel meta-heuristics, and the taxonomy used to structure the presentation. Section "Low-Level Parallelization Strategies" addresses strategies focusing on accelerating computing-intensive tasks without modifying the basic algorithmic design. Methods based on the decomposition of the search space are treated in section "Data Decomposition," while strategies based on the simultaneous exploration of the search space by several independent meta-heuristics constitute the topic of section "Independent Multi--search." Cooperation principles are discussed in section "Cooperative Search" and are detailed in sections "Synchronous Cooperation," "Asynchronous Cooperation," and "Advanced Cooperation Strategies: Creating New Knowledge." We conclude in section "Conclusions."

Meta-heuristics and Parallelism

We start with a brief overview of the potential for parallel computing in meta-heuristics and of performance indicators for parallel meta-heuristics, followed by the criteria used to describe and characterize the parallelization strategies for meta-heuristics described in the other sections of the chapter.

Sources of Parallelism

Addressing a given problem instance with a parallel solution method means that several processes work simultaneously on several processors with the goal of identifying the best solution for the instance. Parallelism thus follows from a decomposition of the total work load and the distribution of the resulting tasks to available processors. According to how "small" or "large" are the tasks in terms of algorithm work or search space, the parallelization is denoted *fine* or *coarse grained*, respectively.

The decomposition may concern the algorithm, the problem instance data, or the problem structure. *Functional parallelism* identifies the first case, where some computing-intensive components of the algorithm are separated into several tasks (processes), possibly working on the "same" data, which are allocated to different processors and run in parallel. The main source of functional parallelism for meta-heuristics is the concurrent execution of their innermost loop iterations, e.g., evaluating neighbors, computing the fitness of individuals, or having ants forage concurrently (section "Low-Level Parallelization Strategies"). This is often also the only source of readily available parallelism in meta-heuristics, most other steps being time dependent and requiring either the computation of the previous steps to be completed or the synchronization of computations to enforce this

time dependency. Consequently, functional parallelism is mainly interesting as a low-level component of hierarchical parallelization strategies or when addressing problem settings requiring a significant part of the computing effort to be spent in the inner-loop algorithmic component.

Parallelism for meta-heuristics may further be found in the domain of the problem addressed or the corresponding search space (for brevity reasons, the term "search space" is used in the rest of the chapter). There are indeed no data dependencies between the evaluation functions of different solutions, and, thus, these may be computed in parallel. Moreover, theoretically, the parallelism in the search space is as large as the space itself. Parallelism is then obtained by separating the search space into components allocated to the available processors. In most cases, these components are still too large for explicit enumeration, and an exact or heuristic search method has to be associated to each to implicitly explore it. *Space separation* is exploited in most of the strategies described in this chapter.

Space separation raises a number of issues with respect to defining an overall meta-heuristic search strategy, in particular, (1) whether to define the separation by partitioning the space, allowing components to partially overlap, or not, (2) how to control an overall search conducted separately on several components of the original space, (3) how to build a complete solution out of those obtained while exploring the components, and (4) how to allocate computing resources for an efficient exploration avoiding, for example, searching regions with poor-quality solutions.

Two main approaches are used to perform the search-space separation: *domain decomposition* (also called *data parallelism*) and *multi-search* (the name *multiple walks* is also found in the literature). The former explicitly separates the search space (section "Data Decomposition") and then addresses the initial problem instantiated on each of the resulting restricted regions, before combining the corresponding partial solutions into complete ones.

Multi-search strategies implicitly divide the search space through concurrent explorations by several methods, named *solvers* in the following. These meta-heuristic or exact solvers may address either the complete problem at hand or explore partial problems defined by decomposing the initial problem through mathematical programming or attribute-based heuristic approaches. In the former case, the decomposition method implicitly defines how a complete solution is built out of partial ones. In the latter case, some processors work on the partial problems corresponding to the particular sets of attributes defined in the decomposition, while others combine the resulting partial solutions into complete solutions to the original problem. Multi-search strategies, particularly those based on cooperation principles, are at the core of most successful developments in parallel meta-heuristics and the object of the largest number of recent publications in the field. Sections "Independent Multi-search," "Cooperative Search," "Synchronous Co-operation," "Asynchronous Cooperation," and "Advanced Cooperation Strategies: Creating New Knowledge" describe multi-search meta-heuristics.

We complete this subsection with a few notes on the performance evaluation of parallel meta-heuristic strategies and resulting algorithms.

The traditional goal when designing parallel solution methods is to reduce the time required to "solve," exactly or heuristically, given problem instances or to address larger instances without increasing the computational effort. For solution methods that run until the provable optimal solution is obtained, this translates into the well-known *speedup* performance measure, computed as the ratio between the wall-clock time required to solve the problem instance in parallel with p processors and the corresponding solution time of the best-known sequential algorithm. A somewhat less restrictive measure replaces the latter with the time of the parallel algorithm run on a single processor. See [9] for a detailed discussion of this issue, including additional performance measures.

Speedup measures are more difficult to interpret when the optimal solution is not guaranteed or the exact method is stopped before optimality is reached. Moreover, most parallelization strategies design parallel meta-heuristics that yield solutions that are different in value, composition, or both from those obtained by the sequential counterpart. Thus, equally important performance measures for parallel heuristics evaluate by how much they outperform their sequential counterparts in (relative) terms of solution quality and, ideally, the computational efficiency. Simply put, the parallel method should not require a higher overall computation effort than the sequential method or should justify the extra effort by higher-quality solutions.

Search robustness is another characteristic expected of parallel heuristics. Robustness with respect to a problem setting is meant here in the sense of providing "equally" good solutions to a large and varied set of problem instances, without excessive calibration, neither during the initial development nor when addressing new problem instances.

Multi-search methods, particularly those based on cooperation, generally offer enhanced performances compared to sequential methods and other parallelization strategies. They display behaviors different from those of the sequential methods involved and can be seen as proper meta-heuristics [2], usually finding better-quality solutions and enhancing the robustness of the meta-heuristic search. See [37,38] for a discussion of these issues.

Characterizing Parallel Meta-heuristic Strategies

Several strategies may be defined based on each one of the sources of parallelism discussed above. The classification of Crainic and Hail [36], generalizing that of Crainic, Toulouse, and Gendreau [43] ([168] and [51] present classifications that proceed of the same spirit), is used in this chapter to characterize these strategies.

The three dimensions of the classification focus on the control of the global problem-solving process, the information exchanged among processes, and the diversity of the search, respectively. The first dimension, *search control cardinality*, thus specifies whether the global search is controlled by a single process or by several processes that may collaborate or not. The two alternatives are identified as *1-control* (*1C*) and *p-control* (*pC*), respectively.

The second dimension addresses the issue of information exchanges and the utilization of the exchanged information to control or guide the search, hence the *search control and communications* name. Communications may proceed either *synchronously* or *asynchronously*. In the former case, processes stop and engage in some form of communication and information exchange at moments (number of iterations, time intervals, specified algorithmic stages, etc.) exogenously planned, either hard-coded or determined by a control (master) process. In the asynchronous communication case, each process is in charge of its own search, as well as of establishing communications with other processes, and the global search terminates once all individual searches stop. Four classes are defined within this dimension to reflect the quantity and quality of the information exchanged and shared, as well as the additional knowledge derived from these exchanges (if any): *rigid (RS)* and *knowledge synchronization (KS)* and, symmetrically, *collegial (C)* and *knowledge collegial (KC)*.

More than one solution method or variant (e.g., with different parameter settings) may be involved in a parallel meta-heuristic. The third dimension thus indicates the *search differentiation* or diversity: do solvers start from the same or different solutions and do they make use of the same or different search strategies? The four classes are *same initial point/population, same search strategy (SPSS)*; *same initial point/population, different search strategies (SPDS)*; *multiple initial points/populations, same search strategies (MPSS)*; and *multiple initial points/populations, different search strategies (MPDS)*. Obviously, one uses "point" for neighborhood-based methods, while "population" is used for genetic-based evolutionary methods, scatter search, and swarm methods.

Low-Level Parallelization Strategies

Functional-parallelism-based strategies, exploiting the potential for task decomposition within the inner-loop computations of meta-heuristics, are often labeled "low level" because they modify neither the algorithmic logic nor the search space. They aim solely to accelerate the search and generally do not modify the search behavior of the sequential meta-heuristic. Typically, the exploration is initialized from a single solution or population and proceeds according to the sequential meta-heuristic logic, while a number of computing-intensive steps are decomposed and simultaneously performed by several processors.

Most low-level parallel strategies belong to the 1C/RS/SPSS class and are usually implemented according to the classical *master-slave* parallel programming model. A "master" program executes the 1-control sequential meta-heuristic, separating and dispatching computing-intensive tasks to be executed in parallel by "slave" programs. Slaves perform evaluations and return the results to the master which, once all the results are in, resumes the normal logic of the sequential meta-heuristic. The complete control on the algorithm execution thus rests with the master, which decides the work allocation for all other processors and initiates most communications. No communications take place among slave programs.

The neighborhood-evaluation procedure of the local search component of neighborhood-based meta-heuristics (as well as of population-based ones implementing advanced "schooling" for offspring) is generally targeted in 1C/RS/SPSS designs. The master groups the neighbors into tasks and sends them to slaves. Each slave then executes the exploration/evaluation procedure on its respective part of the neighborhood and sends back the best, or first improving, neighbor found. The master waits for all slaves to terminate their computations, selects the best move, and proceeds with the search. See, e.g., [70] and [129] for applications of this strategy to tabu search meta-heuristics for the vehicle routing problem with time window constraints (VRPTW) and the scheduling of dependent tasks on heterogeneous processors, respectively.

The appropriate granularity of the decomposition, that is, the size of the tasks, depends upon the particular problem and computer architecture but is generally computationally sensitive to inter-processor communication times and workload balancing. Thus, for example, [54] discusses several decomposition policies for the permutation-based local search neighborhood applied to the scheduling of dependent tasks on homogeneous processors and shows that the uniform partition usually called upon in the literature is not appropriate in this context characterized by neighborhoods of different sizes. The authors also show that a fixed coarse-grained nonuniform decomposition, while offering superior results, requires calibration each time the problem size or the number of processors varies.

The best performing strategy was called by the authors *dynamic fine grained*. It defines each neighbor evaluation as a single task, the master dynamically dispatching these on a first-available, first-served basis to slave processors as they complete their tasks. The strategy partitions the neighborhood into a number of components equal to the number of available processors but of unequal size with a content dynamically determined at each iteration.

The dynamic fine-grained strategy provides maximum flexibility and good load balancing, particularly when the evaluation of neighbors is of uneven length. The uniform distribution appears more appropriate when the neighbor evaluations are sensibly the same, or when the overhead cost of the dynamic strategy for creating and exchanging tasks appears too high.

Similar observations may be made regarding population-based meta-heuristics. In theory, all genetic-algorithm operators may be addressed through a 1C/RS/SPSS design, and the degree of possible parallelism is equal to the population size. In practice, the computations associated to most operators are not sufficiently heavy to warrant parallelizing, while overhead costs may significantly reduce the degree of parallelism and increase the granularity of the tasks. Consequently, the fitness evaluation is often the only target of 1C/RS/SPSS parallelism for genetic-evolutionary methods, the resulting parallel GA being implemented using the master-slave model. Similarly to other 1-control low-level parallelizations, a 1C/RS/SPSS genetic-evolutionary algorithm performs the same search as the sequential program, only faster.

The 1C/RS/SPSS parallelism for ant colony and, more generally, swarm-based methods lies at the level of the individual ants. Ants share information indirectly

through the pheromone matrix, which is updated once all solutions have been constructed. There are no modifications of the pheromone matrix during a construction cycle, and, thus, each individual ant performs its solution-construction procedure without data dependencies on the progress of the other ants.

Most parallel ant colony methods implement some form of 1C/RS/SPSS strategy according to the master-slave model, including [18, 59, 132, 134, 157]. The master builds tasks consisting of one or several ants (each can be assimilated to a "small" colony) and distributes them to the available processors. Slaves perform their construction heuristic and return their solution(s) to the master, which updates the pheromone matrix, returns it to the slaves, and so on. To further speed up computation, the pheromone update can be partially computed at the level of each slave, each slave computing the update associated to its solutions. The fine-grained version with central matrix update has been the topic of most contributions so far, and, in general, it outperformed the sequential version of the algorithm. It is acknowledged, however, that it does not scale, and, similarly to other meta-heuristics, this strategy is outperformed by more advanced multi-search methods.

Scatter search and path relinking implement different evolution strategies, where a restricted number of elite solutions are combined, the result being enhanced through a local search or a full-fledged meta-heuristic, usually neighborhood based. Consequently, the 1C/RS/SPSS strategies discussed previously regarding the parallelization of local search exploration apply straightforwardly to the present context, as in [72–74] for the p-median and the feature selection problems.

A different 1C/RS/SPSS strategy for scatter search may be obtained by running concurrently the combination and improvement operators on several subsets of the reference set. Here, the master generates tasks by extracting a number of solution subsets, which are sent to slaves. Each slave then combines and improves its solutions, returning its results to the master for the global update of the reference set. Each subset sent to a slave may contain exactly the number of solutions required by the combination operator or a higher number. In the former case, the slave performs an "iteration" of the scatter search algorithm [72–74]. In the latter, several combination-improvement sequences could be executed, and solutions could be returned to the master as they are found or all together at the end of all sequences. This heavy load for slaves may conduct to very different computation times, and, thus, load-balancing capabilities should be added to the master.

To conclude, low-level, 1-control parallel strategies are particularly attractive when neighborhoods (populations) are large or neighbor (individual) evaluation procedures are costly, and a significant gain in computing time may be obtained (e.g., the parallel tabu searches of [23, 25, 150] for the quadratic assignment problem (QAP), [24] for the traveling salesman problem (TSP), [128–130] and [54] for the task-scheduling problem). More advanced multi-search strategies generally outperform low-level strategies in most cases. Yet, when a sufficiently large number of processors are available, it might prove worthy to combine a 1C/RS/SPSS approach and more sophisticated strategies into hierarchical solution schemes (e.g., [136] were low-level parallelism accelerated the move evaluations of the individual searches engaged into an independent multi-search procedure for the VRP).

Data Decomposition

Domain or *search-space decomposition* constitutes an intuitively simple and appealing parallelization strategy, dividing the search space into smaller partial sets, solving the resulting subproblems by applying the sequential meta-heuristic on each set, collecting the respective partial solutions, and reconstructing an entire solution out of the partial ones. This apparently simple idea may take several forms, however, according to the type of division performed and the permitted links among the resulting sets/subproblems.

The space may be *partitioned*, yielding disjoint partial sets, or a cover may be defined allowing a certain amount of overlap among partial sets. Thus, for example, the arc-design variables of a VRP may be partitioned into customer subsets (including the depot in each subset), while a cover would allow "close by" customers to belong to two subsets. The goal generally is to generate independent subproblems, which allows to discard from each subproblem the variables and constraints not directly involved in its definition. When this is not possible, e.g., the separated activities share some resources, one may fix the variables not in the subproblem definition (and thus project the corresponding constraints). One then still works on a smaller subproblem, but considering the complete vector of decision variables.

The second element one must consider is the degree of exploration overlap permitted among subproblems. One must thus decide whether the solution transformations (e.g., neighborhood moves or individual crossovers) performed within the partial set of a given subproblem are restricted to that partial set or may involve variables in neighboring subspaces creating an indirect overlapping of subsets. Strict partitioning strategies restrict the solvers to their subsets, resulting in part of the search space being unreachable and the parallel meta-heuristic being nonoptimal. Explicit or implicit overlapping partially addresses this issue. Only partially because, to fully ensure that all potential solutions are reachable, one needs to make overlapping cover the entire search space, which would negate the benefits of decomposition.

Consequently, strict partitioning and very limited overlapping are the preferred approaches found in the literature. A re-decomposition feature is generally included to increase the thoroughness of the search and allow all potential solutions to be examined. The decomposition is thus modified at regular intervals, and the search is restarted using the new decomposition. This feature provides also the opportunity to define a non-exhaustive decomposition, i.e., where the union of the subsets is smaller than the complete search space. A complete solution reconstruction feature is almost always part of the procedure.

This strategy is naturally implemented using master-slave 1C/RS schemes, with MPSS or MPDS search differentiation. The master process determines the decomposition and sends partial sets to slaves, synchronizes them and collects their solutions, reconstructs solutions, modifies the decomposition, and determines stopping conditions. Slaves concurrently and independently perform the search on their assigned partial sets. Design issues one must address in this context are the length of the exploration available to slaves and the reconstruction of global context

information (e.g., global tabu list) out of the partial ones. The extreme case of executing a full meta-heuristic on each partial set of the search space (this avoids the context issue), periodically modifying the partition and restarting the search, was actually generally used, particularly for problems for which a large number of iterations can be performed in a relatively short time, and restarting the method with a new decomposition does not require an unreasonable computational effort (e.g., [66] for the TSP, [95] for image filtering, and [80] for real-time ambulance fleet management).

In a pC/KS strategy, with MPSS or MPDS search differentiation, the decomposition is collegially decided and modified through information exchange phases (e.g., round-robin or many-to-many exchanges) activated at given synchronization points. Such an approach was proposed in [151] for the VRP, where the customer set was partitioned, vehicles were allocated to the resulting regions, and each subproblem was solved by an independent tabu search. All processors stopped after a number of iterations that varied according to the total number of iterations already performed, and the partition was modified by exchanging tours, undelivered cities, and empty vehicles between adjacent processors (corresponding to neighboring regions). At the time, this approach did allow to address successfully a number of problem instances, but the synchronization inherent in the design of the strategy hindered its performance. A parallel ant colony approach combining this decomposition idea to a master-slave implementation was presented in [60] (parallelizing the algorithm presented in [138]), where the master generates an initial solution, defines the partition, and updates the global pheromone matrix, while slaves execute a savings-based ant colony algorithm [137] for the resulting restricted VRP.

Data decomposition methods induce different search behavior and solution quality compared to those of the sequential meta-heuristic. Such methods appear increasingly needed as the dimensions of the contemplated problem instances continue to grow. Given the increased complexity of the problem settings, work is also required on how to best combine search-space decomposition and the other parallelization strategies, cooperation in particular. The integrative cooperative search of [96] is a step in this direction (see section "Advanced Cooperation Strategies: Creating New Knowledge").

Independent Multi-search

Independent multi-search was among the first parallelization strategies proposed in the literature. It is also the most simple and straightforward p-control parallelization strategy and generally offers very interesting performance.

Independent multi-search seeks to accelerate the exploration of the search space toward a better solution (compared to sequential search) by initiating simultaneous solvers from different initial points (with or without different search strategies). It thus parallelizes the multi-start strategy by performing several searches simultaneously on the entire search space, starting from the same or from different initial solutions, and selecting at the end the best among the best solutions obtained by all

searches. Independent multi-search methods thus belong to the pC/RS class of the taxonomy. No attempt is made to take advantage of the multiple solvers running in parallel other than to identify the best overall solution at the synchronization moment when all searches stop.

Independent multi-search turns out to be effective, simply because of the sheer quantity of computing power it allows one to apply to a given problem. Formal insights into the behavior of these strategies may be found in [11, 149, 152, 160]. Empirical efficiency was shown by many contributions that took advantage of its simplicity of implementation and relatively good performance expectation. pC/RS/MPSS parallelizations of neighborhood-based meta-heuristics were thus proposed for, e.g., tabu search for the QAP [11], VRP [136, 152, 156] and production planning [14]; GRASP for the QAP [105, 124, 126], the Steiner problem [110, 111], and the 2-path telecommunication network design [139–141]; simulated annealing for graph partitioning [7, 8, 104] and the TSP [114]; and variable neighborhood search for the p-median problem [71]. Independent multi-search pC/RS/MPSS applications to nongenetic-evolutionary methods have been proposed for scatter search [72, 74], as well as for ant colony optimization for set covering [132], the TSP [149], and the VRP [61]. Similar performance was observed for genetic methods with full-sized populations [28, 29], which avoided the premature convergence observed for pC/RS independent multi-search GA with small-sized populations obtained by separating the initial population among the independent GA searches (e.g., [88, 144]).

Independent multi-search offers an easy access to parallel meta-heuristic computation, offering a tool when looking for a "good" solution without investment in methodological development or actual coding. Independent multi-search methods are generally outperformed by cooperative strategies, however, the latter integrating mechanisms enabling the independent solvers to share, during the search, the information their exploration generates. As explained in the following sections, this sharing and the eventual creation of new information out of the shared one yield in most cases a collective output of superior solutions compared to independent and sequential search.

Cooperative Search

Cooperative multi-search has emerged as one of the most successful meta-heuristic methodologies to address hard optimization problems (see, e.g., [2, 32, 33, 36, 39, 40, 155, 159]). Cooperative search is based on harnessing the capabilities of several solution methods through *cooperation mechanisms* providing the means to share information while addressing the same problem instance (and create new information out of the exchanged data in advanced settings; see section "Advanced Cooperation Strategies: Creating New Knowledge").

Cooperative search strategies are thus defined by the solver components engaged in cooperation, the nature of the information shared, and their interaction mechanism. The solvers define trajectories in the search space from possibly different

initial points or populations, by using possibly different search strategies (including the same method with different parameter settings or populations). The information-sharing cooperation mechanism specifies how these independent solvers interact, how the exchanged information is used globally (if at all), and how each process acts on the received information, using it within its own search and, thus, transforming it before passing it to other solvers. As further detailed in the following sections, various cooperation mechanisms have been proposed: diffusion among "neighboring" solution methods arrayed in particular communication architectures, e.g., fine-grained, cellular GA (e.g., [21, 108]) and multilevel cooperative search [165]; direct exchanges among processes as in coarse-grained, island GA [21, 108], A-teams [158, 159], and collegial asynchronous strategies [42, 43]; and indirect exchanges through a common data repository and management structure such as the *adaptive* [6, 12, 142] and *central memory* [42, 43, 46, 93, 94, 100] strategies. The global search behavior of the cooperative parallel meta-heuristic then emerges from the local interactions among the independent solvers, yielding a "new" meta-heuristic in its own right [39]. The similarity between this behavior and that of systems where decisions emerge from interactions among autonomous and equal "colleagues" has inspired the name *collegial* associated to cooperative search strategies in the taxonomy used in this chapter.

The exchanged information has to be *meaningful* and *timely*. The goals are twofold. First is to enhance the performance of the receiving solvers. Second is to create a global image of the status of the cooperative search as "complete" as possible, which would provide the means to guide the global search toward better performance in terms of computing time and solution quality than the simple concatenation of the results of the individual solvers. Of course, one desires to achieve these goals without excessive overhead. Toulouse, Crainic, and Gendreau [162] proposed a list of fundamental issues to be addressed when designing cooperative parallel strategies for meta-heuristics: What information is exchanged? Between what processes is it exchanged? When is information exchanged? How is it exchanged? How is the imported data used? Implicit in their taxonomy and explicitly stated in later papers, the issue of whether the information is modified during exchanges or whether new information is created completes this list.

"Good" solutions make up the most often exchanged type of information. This usually takes the form of the local current best solution of a given solver or the overall best. The question of when to send such solutions has to be carefully addressed, however, particularly when the receiving process is supposed to act momentarily on the incoming information. One should thus avoid sending all local current best solutions, particularly when the solver is performing a series of improving moves or generations, as successive solutions are generally "similar" and the receiving solvers have no chance to actually act on the incoming information. Sending the overall current best solution to all cooperating solvers should also be avoided, as it rapidly decreases the diversity of the exploration and, thus, increases the amount of worthless computational work (many solvers will search within the same region) and brings an early "convergence" to a not-so-good solution. Sending out local optima only, exchanging groups of solutions, implementing randomized

selection procedures (generally biased toward good or good-and-diverse solutions) of the solutions to share, and having the cooperating solvers treat differently the received information are among the strategies aimed at addressing these issues.

Context information may also be profitably shared and integrated into the mechanisms used to guide the overall search. Context information is routinely collected by meta-heuristics during their search. It may consist in statistical information relative to the presence of particular solution elements in improving solutions (e.g., the medium- and long-term memories of tabu search), the impact of particular moves on the search trajectory (e.g., the scores of the moves of large adaptive neighborhood search), population diversity measures, individual resilience across generations, etc. A limited number of studies indicate the interest of context information exchanges (see section "Advanced Cooperation Strategies: Creating New Knowledge"), but more research is needed on this topic.

Cooperating solvers may communicate and exchange information directly or indirectly. *Direct* exchanges of information occur either when the concerned solvers agree on a meeting point in time to share information or when a solver broadcasts its information to one or several other solvers without prior mutual agreement. The latter case is generally not performing well, except when solvers include costly mechanisms to store such information without disturbing their own execution until ready to consider it.

Indirect exchanges of information are performed through independent data structures that become shared sources of information solvers may access according to their own internal logic to post and retrieve information. Such data structures are known under various names, e.g., *blackboard* in computer science and artificial intelligence vocabulary and *memory*, *pool*, and *data warehouse* (the terms *reference* and *elite set* are also sometimes used) in the parallel meta-heuristic literature. The term *memory* is used in the following.

Centralized memory is the implementation of choice reported in the literature. Distributed memory mechanisms may be contemplated, where a number of memories are interconnected, each servicing a number of solvers. Such hierarchical structures, involving several layers of solvers and memories, appear interesting when a large number of processors are to be involved, for integrative cooperation strategies, or when computations are to take place on grids or loosely coupled distributed systems. Issues related to data availability, redundancy, and integrity must be then addressed, as well as questions relative to the balancing of workloads and the volume of information exchanged. More research is needed on this topic.

The logical intercommunication network corresponding to the selected cooperation strategy takes the form of a *communication graph*. A node of the graph represents a solver or a memory. Edges define the pairs of solvers or of a solver and a memory that may communicate directly. The projection of the communication graph on the physical interconnection topology of the parallel computer executing the parallel program – complete graph, ring, grid, torus, and star are most often encountered in the literature – is normally part of the implementation process.

When and how information is exchanged specifies how frequently cooperation activities are initiated, by whom, and whether all concerned solvers must

simultaneously engage in communications or not. *Synchronous* and *asynchronous* communications are the two classes of communication exchanged and are discussed in the following sections. The accumulated knowledge of the field indicates for both classes that exchanges should not be too frequent to avoid excessive communication overheads and premature "convergence" to local optima [42, 43, 162].

Three remarks complete this section. First, "simple" cooperation designs based, for example, on synchronization or on exchanging current best solutions only often appear biased toward intensifying the search in already-explored regions where interesting solutions have been identified. Diversification capabilities, e.g., probabilistic or diversity-driven selection of exchanged solutions, are thus an important component of cooperative p-control strategies.

One also observes that the main principles of cooperative parallel strategies are the same for neighborhood- and population-based meta-heuristics, even though denominations and implementation approaches may differ. The terms *coarse-* and *fine-grained island* are thus used to identify the amplitude of the population (large or small, down to single individual eventually, respectively) of participating solvers in genetic-based cooperation. Similarly, *multi-colony* is the term generally used for cooperation in the ant colony community. The next sections are thus structured around classes of strategies rather than by meta-heuristic type.

Finally, one should notice that cooperation takes place at two different levels. The first is the *explicit* information sharing specified by the design of cooperation mechanism. *Implicit* cooperation makes up the second level, where information spreads across the cooperating solvers through a diffusion process and correlated interactions [163, 164, 166, 167]. Implicit cooperation is **not** specified explicitly in the design of the algorithm. It is thus a bottom-up, global emergent phenomenon produced by the correlated interactions among searches. Important research issues and challenges are related to how to harness indirect cooperation to enhance the optimization capabilities of cooperative search. For example, how should one select solvers and direct cooperation mechanisms to yield a system-wide emergent behavior providing an efficient exploration of the solution space from an optimization point of view? Learning and dynamic self-adaptation, at the level of both individual solvers and the cooperating meta-heuristic, appear to be part of the answer. Several other areas of research study systems displaying emergent behaviors, e.g., decentralized autonomic computing, social robotics, swarm intelligence, clustering logistics activities in supply chains, etc., and cross-fertilization appear promising. Empirical and theoretical research in this area should yield design principles and tools for more powerful (parallel) meta-heuristics.

Synchronous Cooperation

Synchronous cooperation follows a pC/KS scheme, with any of the SPDS, MPSS, or MPDS search differentiation approaches, according to which the independent cooperating meta-heuristics synchronize at particular moments to initiate an information exchange phase. Synchronize here means that every solver but the last stops

its activities and waits for all others to be ready. The synchronization moments may be generated based on conditions external to all solvers (e.g., number of iterations since the last synchronization) or detected by a specially designated solver. The information exchange phase must be completed before any solver can restart its exploration from the respective synchronization point.

Synchronization may use a complete communication graph or a more restricted, less densely connected communication topology, e.g., a ring, torus, or grid graph. *Global* exchanges of information among all solvers take place in the former case, while information follows a *diffusion* process through direct *local* exchanges of information among neighboring processes in the latter. In all cases, the objective is to re-create a state of complete knowledge at particular moments in the global search and, thus, to hopefully guide it toward a coordinated evolution to the desired solution to the problem. This goal is rarely attained, however, and the price in computing-time efficiency may be significant, as communications cannot be initiated before the slowest solver is ready to proceed.

Global Information Exchanges

Many pC/KS cooperative search meta-heuristics in the literature implement the strategy according to the master-slave model. The master process, which may or may not include one of the participating solvers, initiates the other processes, stops them at synchronization points, gathers the information to be shared, updates the global data, decides on the termination of the search and, either effectively terminates it or distributes the shared information (a good solution, generally, the overall best solution in many cases) to the solvers for the continuation of the search.

The VNS pC/KS method for the p-median problem proposed in [71] followed this idea, as well as the tabu search-based algorithms proposed for the TSP [109], the VRP (using ejection chains) [135, 136], the QAP [55] and the task mapping problem [56], the last two contributions attempting to overcome the limitations of the master-slave implementation by allowing processes, on terminating their local search phases, to synchronize and exchange best solutions with processes running on neighboring processors (this idea represents a step toward a "true" pC/KS design using a partial solution-diffusion process). This idea was also used to implement coarse-grained island-based cooperating genetic methods [52, 148], where the master stopped the cooperating meta-heuristics to initiate a *migration* operator exchanging among the independent populations the best or a small subset of the best individuals in each. Applied to ant colony systems [64], this strategy divided the colony into several subcolonies, each assigned to a different processor. Each independent ant colony meta-heuristic sent to the master its best solution once its ants finished searching. The master updated the pheromone matrix and started a new search phase. A more sophisticated pC/KS approach was proposed in [120] for the 0–1 multidimensional knapsack problem, where the master dynamically adjusted the parameters of the cooperating tabu searches according to the results each had obtained so far. Computational results showed this dynamic adjustment to be beneficial.

Alternatively, pC/KS schemes can be implemented in "true" collegial fashion by empowering each cooperating solver to initiate synchronization once it reaches a predetermined status. It then broadcasts its data, followed by similar broadcasts performed by the other solvers. Once all broadcasts are completed and information is shared, each solver performs its own import procedures on the received data and proceeds with its exploration of the search space until the next synchronization event.

Such an approach, exchanging the current best solution or group of solutions, was proposed for simulated annealing [58], where the solvers transmitted their best solutions every n steps and restarted the search after updating their respective best solutions (see also [101–104] for the graph partitioning problem). For tabu search applied to location problems with balancing requirements [41, 42], solvers synchronized after a number of iterations either predefined or dynamically determined. Most synchronous coarse-grained island genetic parallel methods applied this strategy, migration operators being applied at regular intervals, e.g., [171] for satisfiability problems (the best individual of each population migrated to replace the worst of the receiving population), [67] for multi-objective telecommunication network design with migration following each generation, and [27–29, 89, 107] for graph partitioning, the latter implementing a hierarchical method, where the fitness computation was performed at the second level (through a master-slave implementation; the overhead due to the parallelization of the fitness became significant for larger numbers of processors). A similar strategy was proposed for the multi-ant colony algorithms [112, 113]. Each colony has its own pheromone matrix and may (homogeneous) or may not (heterogeneous) use the same update rule. Colonies synchronize after a fixed number of iterations to exchange elite solutions that are used to update the pheromone matrix of the receiving colony.

Synchronization involved the exchange of not only good solutions but also of important search parameters in the pC/RS/MPDS parallel iterated tabu search proposed for the vehicle routing problem (VRP) [30]. The iterated tabu solvers started from different initial solutions and used different search parameters. They synchronized based on the number of consecutive iterations without improvement used to determine the stopping moment of the individual improvement phases. This provided the means to more equally distribute the work among cooperating processes. The solvers exchanged their best solutions, each solver probabilistically selecting the working solution for the next improvement phase among the received ones and its own. This method proved to be both flexible and efficient for several classes of routing problem settings with several depots, periodicity of demands, and time windows.

Most studies cited above compared several parallel strategies for the metaheuristic and problem setting at hand [27–29,41,42,101–104,107]. They contributed to show that synchronous pC/KS strategies with global information exchanges outperform independent search approaches, as well as the respective sequential version, particularly with respect to solution quality. These studies also pointed out, however, the benefit of dynamically determined synchronization points, as well as the superiority of asynchronous communications.

Diffusion

The previous strategies are based on global exchanges of information, gathered at synchronization points during the computation and distributed to all search threads. The interest of global information-sharing strategies resides in the best information available at the synchronization moment being available to all the solvers involved in cooperation. The main drawback results from this same characteristic, however, as solvers relying heavily on the same information (a set of best solutions in most cases) tend to focus on the same regions of the search space. This generally results in a search lacking in diversity that, more often than not, proves inefficient.

Synchronized cooperation strategies based on *diffusion* of information through local exchanges among "neighboring" solvers have therefore been proposed. Such approaches are defined on sparse communication graphs displaying particular topologies, such as ring, torus, or grid graphs, where each node is directly linked to only a few other nodes. A solver is then a neighbor of another solver if there is a direct link between the two nodes on which they run, that is, if their nodes are adjacent in the communication graph.

Synchronization still means that all solvers stop and exchange information, but here they perform it with their neighbors exclusively. Consequently, the quantity of information each solver processes and relies upon is significantly reduced, while the exchanges between nonadjacent solvers are performed at the speed of diffusion through possibly several chains of local exchanges and data modifications.

This idea has been much less explored compared to the global-exchange strategy, even though synchronous cooperative mechanisms based on local exchanges and diffusion have a less negative impact on the diversity of the search-space exploration. A number of applications were proposed with good results for coarse-grained [19,161] and fine-grained [3,63,68,69,117,118] genetic-based evolutionary methods, as well as for ant colony optimization [113].

Cooperation based on asynchronous information sharing generally outperforms synchronous methods, however, and is the topic of the next subsection.

Asynchronous Cooperation

Asynchronous strategies largely define the "state of the art" in parallel multi-search meta-heuristics. Solvers initiate asynchronous cooperation activities according to their own design logic and current internal state only, without coordination with other solvers or memories. Information sharing then proceeds either by direct inter-solver exchanges or through a data repository structure. These strategies belong to either the pC/C, described in this section, or the pC/KC, described in the next section, collegial classes of the taxonomy, the latter using the shared information to generate new knowledge and global search guidance.

Two main benefits are obtained when relying on asynchronous communications. First, this provides the means to build cooperation and information sharing among solvers without incurring the overheads associated to synchronization. Second, it increases the adaptability of cooperative meta-heuristics, as their capability to react

and dynamically adapt to the exploration of the search space by the cooperating solvers is significantly increased compared to the other parallelization strategies. Of course, these benefits come with potential issues one must care for. For example, the information gathered during the search will seldom, if ever, be completely available when a process must decide. Also, too frequent data exchanges, combined to simple acceptance rules for incoming information, may induce an erratic solver behavior, the corresponding search trajectories becoming similar to random walks. Hence the interest for applying information-sharing quality, meaningfulness, and parsimony principles [42, 43, 162].

In the basic asynchronous strategies discussed in this section, the shared information generally corresponds to a locally improving solution or individual(s). Most successful implementations have their cooperating solvers send out new local optima only. This limits the quantity of information sent and received, as well as the amount of work required to process it. Moreover, it avoids the case where a solver reorients its search based on one of a series of improving solutions and ends up developing a trajectory similar to the one followed by the solver that originated the data.

The abovementioned principles also explain the interest in diversifying the shared information [42]. Thus, always selecting the best available solution out of an elite set of good solutions, sent by potentially different solvers, proved less efficient in terms of quality of the final solution than a strategy that randomly, biased by quality, selected among the same elite set.

Finally, when to initiate cooperation activities and how to use incoming information is particular to each type of meta-heuristic involved in the cooperation. Yet, common to most strategies proposed in the literature is to perform jointly the sending and requesting of information. There is no absolute need to do this, however, even though it might decrease the amount of communication work. It might thus be interesting for neighborhood-based methods to make available right away their newly found local optima or improved overall solution and not wait for the algorithmic step where examining external information is appropriate. Similarly, population-based methods could migrate a number of individuals when a significant improvement is observed in the quality and diversity of their elite group of individuals.

With respect to when to request external information, the parsimony principle implies selecting only moments when the status of the search changes significantly, such as when the best solution or the elite subpopulation did not improve for a number of iterations.

The solver then engages into a so-called *search-diversification* phase, e.g., diversification in tabu search, change of neighborhood in variable neighborhood search, and complete or partial regeneration of population in population-based meta-heuristics, involving the choice or modification of the solution to initiate the new phase. Examining the contribution of external information is appropriate in this context. Notice that it is always possible to use simply a prefixed number of iterations to initiate communications, but this approach should be restricted to meta-heuristics without search-diversification steps, e.g., tabu search based on continuous diversification.

Direct-exchange strategies are generally implemented over a complete communication graph, each solver sending out information to all other solvers or to a subset of them; this subset may be predefined or selected dynamically during the search. Particular communication graphs and information-diffusion processes could also be used but, despite encouraging results, too few experiments have been reported yet (e.g., [146] proposing VNS pC/C strategies over uni- and bidirectional ring topologies). Each solver was executing the basic VNS steps and, on competing them, was passing its solution to its next neighbor (uni) or its two neighbors (bi), while receiving a solution from its predecessor neighbor (uni) or its two neighbors (bi). The received solution was kept as initial solution of the next VNS run in the unidirectional case, while the best of the two received ones and the local one was kept in the bidirectional ring implementation. The latter strategy proved the most successful.

Information exchanges within pC/C strategies based on indirect communications are generally performed through a data repository structure, often called *central memory* [32, 42, 43]. A solver involved in such a cooperation deposits (sends) good solutions, local optima generally, into the central memory, from where, when needed, it also retrieves information sent by the other cooperating solvers. The central memory accepts incoming solutions for as long as it is not full, acceptance becoming conditional to the relative interest of the incoming solution compared to the "worst" solution in the memory, otherwise. Evaluation is performed using the evaluation function for the global search space (or the objective function of the original problem). Diversity criteria are increasingly considered is this evaluation, a slightly worst solution being preferred if it increases the diversity of solutions in the central memory. Population culling may also be performed (deleting, e.g., the worst half the solutions in memory).

Both approaches may be applied to any meta-heuristic but, historically, most pC/C genetic-based evolutionary asynchronous cooperative meta-heuristics implemented a coarse-grained island model with direct inter-solver exchanges. An early comparative study of coarse-grained parallel genetic methods for the graph partitioning problem numerically showed the superiority of the pC/C strategy (with migration toward a subset of populations) over synchronous approaches [107].

The indirect-exchange communication model is found at the core of most asynchronous cooperative search strategies outside the genetic-evolutionary community, including simulated annealing for graph partitioning [101–104] and the TSP [143] and VNS applied to the VRPTW [127] and the p-median problem [44]. A master process was associated to the central memory in the latter method, which kept, updated, and communicated the current overall best solution (it also initiated and terminated the algorithm). Individual solvers proceeded with the VNS exploration for as long as the solution was improved. When this was no longer the case, the current best was communicated to the master (if better than the one at the last communication), and the overall best solution was requested from it. The best solution between the local and imported ones was selected, and the search was then continued in the current (local) neighborhood. Computational results on TSPLIB problem instances with up to 11,849 customers showed that the cooperative strategy

yielded significant gains in terms of computation time without losing on solution quality.

Apparently, [42] proposed the first central memory asynchronous tabu search. The tabu search solvers addressed a multicommodity location problem with balancing requirements. Each solver sent to the memory its local-best solution when improved and imported a probabilistically selected (rank-biased) solution from the memory before engaging in a diversification phase. This method outperformed in terms of solution quality the sequential version, several synchronous variants, and a broadcast-based asynchronous pC/C cooperative strategy. The same approach was applied to the fixed cost, capacitated, multicommodity network design problem with similar results [35]. Similar approaches were proposed for a broad range of problem settings, including the partitioning of integrated circuits for logical testing [1], two-dimensional cutting [13], the loading of containers [15], labor-constrained scheduling [22], the VRPTW [99], and the capacitated VRP [93].

Solvers involved in pC/C strategies may not be restricted to a single metaheuristic. Thus, the solvers in the two-phase approach of [75–77,91] for the VRPTW first applied an evolution strategy to reduce the number of vehicles, followed by a tabu search to minimize the total distance traveled. A different version of the same idea may be found in [10] for the Steiner problem, where each phase of the two phase is designed as a pC/C asynchronous central memory strategy, only the change from one phase to the next being synchronized. Solvers run reactive tabu search and path relinking meta-heuristics in the first and second phases, respectively.

Multilevel cooperative search proposes a different pC/C asynchronous cooperative strategy based on controlled diffusion of information [165]. Solvers are arrayed in a linear, conceptually vertical, communication graph, and a local memory is associated to each. Each solver works on the original problem but at a different level of aggregation (the solver on the conceptually first level works on the complete original problem) and communicates exclusively with the solvers directly above and below, that is, at higher and lower aggregation levels, respectively. The local memories are used to send information to the immediate neighbors and to access the incoming data from the same, at moments dynamically determined according to the internal logic of the respective solver. In the original implementation, solvers were exchanging improved solutions, incoming solutions not being transmitted further until modified locally for a number of iterations to enforce the controlled diffusion of information. Excellent results have been obtained for various problem settings including graph and hypergraph partitioning problems [122, 123], network design [47], feature selection in biomedical data [121], and covering design [53]. It is noteworthy that one can implement multilevel cooperative search using a central memory by adequately defining the communication protocols. Although not yet fully defined and tested, this idea is interesting as it opens the possibility of richer exchange mechanisms combining controlled diffusion and general availability of global information.

The central memory pC/C asynchronous cooperation strategy has proved worthy by several criteria. It yields high-quality solutions and is computationally efficient as no overhead is incurred for synchronization. It also helps to address the issue

of "premature convergence" in cooperative search, by diversifying the information received by the participating solvers through probabilistic selection from the memory and by a somewhat large and diverse population of solutions in the central memory (solvers may thus import different solutions even when their cooperation activities are taking place within a short time span).

The performance of central memory cooperation and the availability of exchanged information (kept in the memory) have brought the question of whether one could design more advanced cooperation mechanisms taking advantage of the information exchanged among cooperating solvers. The pC/KC strategies described in the next section are the result of this area of research.

Advanced Cooperation Strategies: Creating New Knowledge

Cooperation, in particular, memory-based asynchronous cooperation, offers a rich framework to combine solvers of different meta-heuristic and exact types, together with a population of elite solutions of continuously increased quality. But, is the development effort worthwhile?

An interesting proof of concept is found in the study of Crainic and Gendreau [34] combining a genetic-method solver and an asynchronous pC/C tabu search for multicommodity location-allocation with balancing requirements [42]. The tabu search solvers were aggressively exploring the search space, building the elite solution population in the central memory. The genetic method initialized its population with the one in the central memory once it contained a certain number of solutions. Its aim was to create new solutions to hopefully enrich the quality and diversity of the solutions exchanged among the cooperating solvers. Asynchronous migration transferred the best solution of the genetic population to the central memory, as well as solutions from the central memory toward the genetic population. This strategy did perform well, especially on larger instances. Most importantly, it showed that, while the overall best solution was never found by the genetic solver, the GA-enhanced cooperation yielded higher-quality solutions compared to the cooperation involving the tabu searches only. It appeared that the newly created solutions offered a more diverse set of diversification opportunities to the tabu search solvers, translating into a more diverse global search yielding better solutions.

The conclusion of that paper was not only that it is worthwhile to involve solvers of different types in the cooperation but also that it is beneficial to create new solutions out of the set of elite solutions in the central memory. The new solutions are different from their parent solutions and are added to the central memory if they improve compared to them. The process is thus doubly beneficial as better solutions in the memory directly enhance the quality of the global search, while increasing the diversity of solutions in memory provides the opportunity for cooperating solvers to explore new regions of the search space.

A second idea on developing advanced cooperation mechanisms concerns the information that may be extracted from the exchanged solutions, and the context information, eventually. It has thus been observed that optimal or near-optimal

solutions are often similar in the values taken by a large number of variables. Moreover, it is well known in the meta-heuristic field that one can learn from the characteristics of the solutions generated during the search, out of the best ones in particular, and use this learning to guide the search (see, e.g., the studies on memory and learning performed for tabu search [82]). Applied to cooperative search, it appeared promising to apply these learning techniques to the elite solutions in the population gradually built in the central memory and to use the resulting information to guide the search performed by the cooperating solvers.

Asynchronous cooperative strategies that include mechanisms to create new solutions and to extract information out of the exchanged solutions make up the p-control knowledge collegial (pC/KC) class. In most developments in this field, cooperating solvers work on the complete problem formulation and data. A recent research trend addresses rich multi-attribute problem settings and proposes pC/KC strategies where different solvers work on particular parts of the initial problem or on integrating the resulting partial solutions into complete ones. The next subsections describe these two cases.

pC/KC with Solvers Working on the Complete Problem

Two main classes of pC/KC cooperative mechanisms are found in the literature differing in the information that is kept in memory. *Adaptive memory* methods store partial elements of good solutions [142], while complete ones are kept in *central memory* methods [32, 38, 42]. The latter method generalizes the former and the vocabulary used in the various papers notwithstanding the two approaches is becoming increasingly unified.

The *adaptive memory* terminology was coined by Rochat and Taillard [142] (see also [81, 153, 154]). The method was targeting the VRP and the VRPTW, and it marked a changing point in the state of the art at the time. The main idea was to keep in the memory the individual components (vehicle routes in the initial application) of the solutions produced by the cooperating solvers (tabu search methods in [142]). Two types of information were recorded for each solution element kept in memory, a frequency counter of its appearance in the best solutions encountered so far, and its rank within the elite population in memory based on the characteristics (mainly the objective function value) of the solution from which it was extracted. Solvers constructed new complete solutions out of randomly (rank-biased) selected partial elements, improved these new solutions, and returned the best ones to the memory. The rank-biased random selection of elements assured that the new solution was composed in most cases of routes from different elite solutions, thus inducing a powerful diversification effect.

Several interesting developments were proposed, and conclusions were drawn within the context of successful adaptive memory pC/KC algorithms. A set-covering heuristic was thus proposed as selection mechanism for the elements (VRPTW routes) used by solvers to generate new initial solutions [145]. This mechanism proved very successful and has been used several times since (e.g., [87]). A

two-level parallel scheme was proposed for the real-time vehicle routing and dispatching [79]. A pC/KC/MPSS cooperating adaptive memory method made up the first level, while the slave processors attached to each solver, a tabu search method based on route decomposition [151], made up the second level. The performance of this method is noteworthy also because, while many papers mention the possibility of hierarchical parallel schemes, very few actual implementations are found in the literature. Equally for the VRPTW, the adaptive memory approach of [6] yielded a number of interesting findings relative to the implementation of cooperative methods. Thus, when individual solvers are fast, as is generally the case for routing problems, it is beneficial to run several solvers on the same processor and group the exchanges with the central adaptive memory (avoiding or reducing access bottlenecks to the latter). On the other hand, running the memory and solvers on the same processor is to be avoided (the solver execution reduces the response efficiency of the memory).

Solvers in *central memory* methods indirectly exchange complete elite solutions and context information through the central memory data repository device. Solvers may include constructive, improving, and post-optimization heuristics (e.g., [99, 100]), neighborhood (e.g., tabu search [57, 93, 94]) and population-based methods (e.g., genetic algorithms [57, 99, 100], and path relinking [46]), as well as exact solution methods, on restricted versions of the problem, eventually. The particular solvers to include depend on the particular application. They should be efficient for the problem at hand. They should also contribute to build and enhance solutions that may contribute to improve both the quality and the diversity of the elite population being built in the central memory.

The central memory keeps full solutions, solution attributes and context information, both received from the solvers and newly created out of the exchanged information. To more clearly distinguish between the data warehousing and the information creating functions of central memory mechanisms, let the *search coordinator* (*SC*) be the process in charge of the latter function. The simplest version of the SC was used in the pC/C strategies of the previous section, where solutions in memory were ordered (generally according to the value of the objective function) and rank-biased randomly extracted to answer solver requests. The functions of the SC in pC/KC methods include creating new solutions; extracting appropriate solution elements; building statistics on the presence and performance of solutions, solution elements, and solvers (these belong to the family of memories well known in the meta-heuristic community); and creating the information to return when answering solver requests (the latter are the so-called *guidance mechanisms*).

The cooperative meta-heuristic proposed by [99] for the VRPTW used a simple pC/KC mechanism. Four solvers, two simple genetic algorithms with order and edge recombination crossovers, respectively, and two tabu search methods that perform well sequentially, the unified tabu [31] and TABUROUTE [78]. The solvers sent their improved best solutions to the central memory and requested solutions from the same when needed (the genetic algorithms for crossover operations, at regular intervals for the unified tabu, and at diversification time for TABUROUTE). Besides ordering and selecting the solutions to return, the SC was only performing

post-optimization (2-opt, 3-opt, or-opt, and ejection-chain procedures to reduce the number of vehicles and the total traveled distance) on the received solutions. Without any calibration or tailoring, this algorithm proved to be competitive with the best meta-heuristics of its day in linear speedups.

A more complete SC was proposed in [100] also for the VRPTW. The goal was for a guidance mechanism that, first, extracted and returned to solvers meaningful information in terms of individual guidance and global search performance and, second, was independent of problem characteristics, routes in particular, and could be broadly applied to network-based problem settings. To work toward the second goal, the SC mechanism targeted an atomic element in network optimization, the arc. The basic idea of the SC mechanism was that an arc that appears often in good solutions and less frequently in bad solutions may be worthy of consideration for inclusion in a tentative solution, and vice versa. To implement this idea, the authors considered the evolution of the "appearance" of arcs in solutions of different qualities. Appearance was measured by means of frequencies of inclusion of arcs in the elite (e.g., the 10 % best), average (between the 10 % and 90 % best), and worst (the last 10 %) groups of solutions in the central memory. *Patterns* of arcs were then defined representing subsets of arcs (not necessarily adjacent) with similar frequencies of inclusion in particular population groups. Guidance was obtained by transmitting arc patterns to the individual solvers indicating whether the arcs in the pattern should be "fixed" or "prohibited" to intensify or diversify the search, respectively. The solvers accounted for the "fix" and "prohibit" instructions by using the patterns to bias the selection of arcs for move or reproduction operations. A four-phase fixed schedule (two phases of diversification at the beginning to broaden the search, followed by two intensification phases to focus the search around promising regions) was used (see [98] for a dynamic version of this mechanism). Excellent performances in terms of solution quality and computing efficiency were observed compared to the best performing methods of the day.

A different SC was proposed in [94] for the capacitated VRP with tabu search solvers. Solvers periodically (after a number of iterations or when the solution has not been improved for a number of iterations) sent best solutions to central memory, and received a solution back from it, the search being resumed from the received solution. The SC mechanism aimed to identify and extract information from the solutions in memory to guide solvers toward intensification and diversification phases. This was obtained by clustering solutions, dynamically when solutions were received, according to the number of edges in common. Thus, solutions in a given cluster have a certain number of edges in common, this cluster of edges and solutions being assumed to represent a region of the search space. Search history indicators were also associated to clusters giving the number of solutions in the cluster and the quality of the solutions. This information was used to infer how thoroughly the corresponding region had been explored and how promising it appeared. Clusters were actually sorted according to the average solution value of their feasible solutions. The cluster with the lowest average value, that is, with a largest number of very good solutions, was selected for intensification, while the solution with the lowest number of good solutions was selected for diversification.

A solution was selected in the corresponding cluster, and it was sent to the requesting solver. Excellent results were obtained in terms of solution quality and computation effort (an almost linear speedup was observed up to 240 processors) compared to the state-of-the-art methods of the day (including the parallel method of [87]).

A pC/KC/MPDS method proposed in [87] for the VRP demonstrates how specialized solvers may address different issues in a cooperative meta-heuristic, including the generation of new knowledge. Two types of solvers were defined in this scheme. The so-called heuristic solvers improved solutions received from the SC associated to the central memory (called master in [87]), through a record-to-record meta-heuristic [26, 83, 106]. On completing the task, solvers returned both a number (50) of the best solutions found and the corresponding routes (a post-optimization procedure was first run on each route). Simultaneously, set-covering solvers aimed to identify new solutions by solving a series of set-covering problems starting from a limited set of routes. Each time a set-covering problem was solved, the solution was returned to the central memory and the set of the current ten best solutions was retrieved for the next run. Set-covering solvers had also access to the ordered list of best routes in memory, and they selected within to complete their problems. The number of routes admitted to set up a set-covering problem was dynamically modified during the search to control the corresponding computational effort. The SC kept and ordered the received solutions and routes and selected the solutions to make available to solvers (routes were always available; an efficient file system was used to facilitate access to this data). The method performed very well, both in terms of solution quality and computational effort (an almost linear speedup was observed).

The contributions described in this section emphasize the interest of asynchronous knowledge-generating cooperative meta-heuristics. The cooperation and guidance mechanisms, as well as the role of learning and statistical performance data, require additional and systematic studies, preferably on a broader range of problem settings. The contributions aimed at addressing multi-attribute problem settings are described in the next subsection.

pC/KC with Partial Solvers: The Integrative Cooperative Search

The versatility and flexibility of the central memory concept has raised the interest in generalizing it to address so-called *rich* combinatorial optimization problems displaying a large number of attributes characterizing their feasibility and optimality structures. The general idea is to decompose the initial problem formulation along sets of decision variables, called *decision-set attribute decomposition* in [96], yielding simpler but meaningful problem settings, in the sense that efficient solvers, can be "easily" obtained for these partial problems either by opportunistically using existing high-performing methods or by developing new ones. The central memory cooperative search framework then brings together these partial problems and their associated solvers, together with integration mechanisms, reconstructing complete solutions, and search-guidance mechanisms.

The first effort in this direction is probably the work of [46] (see also [57]) for the design of wireless networks, where seven attributes were considered simultaneously. The proposed pC/KC/MPDS cooperative meta-heuristic had tabu search solvers work on limited subsets of attributes only, while a genetic method amalgamated the partial solutions sent by the tabu search solvers to the central memory, into complete solutions to the initial problem.

The general method, called *integrative cooperative search ICS*) by its authors, has been fully defined in [96] (see also [48,49]). The brief presentation that follows describes ICS according to [96] through an application to the multi-depot periodic vehicle routing problem (MDPVRP), which simultaneously decides on (1) selecting a visit *pattern* for each customer, specifying the particular periods the customer is to be visited over the multi-period planning horizon, and (2) assigning each customer to a depot for each visit [115, 169].

The main components of ICS, which must be instantiated for each application, are the (1) decomposition rule; (2) *partial solver groups* (*PSGs*) addressing the partial problems resulting from the decomposition; (3) *integrators*, which select partial solutions from PSGs, combine them to create complete ones, and send them to the *complete solver group* (*CSG*); and (4) the CSG, which corresponds to the central memory of ICS and has as prime function to manage the pool of complete solutions and the context information received from the PSGs and integrators and to extract out of these the information required to guide the partial and global searches. Guidance is performed by the *global search coordinator* (*GSC*) associated to the CSG. Notice that, in order to facilitate the cooperation, a unique solution representation is used throughout ICS. This representation is obtained by fixing rather than eliminating variables when defining partial problems.

The selection of the decision sets is specific to each application case, decision variables being clustered to yield known or identifiable optimization problem settings. An opportunistic selection decomposes the MDPVRP along the depot and period decision sets to create two partial problems. Thus, fixing the customer-to-depot assignments yields a periodic VRP (PVRP), while fixing the patterns for all customers yields a multi-depot VRP (MDVRP). High-quality solvers exist in the literature for both problems.

Two PSGs were defined for the partial problems, one for the PVRP and the other for the MDVRP. Each PSG was organized according to the pC/KC paradigm and was thus composed of a set of *partial solvers*, a central memory where elite solutions were kept, and a *local search coordinator* (*LSC*) managing the central memory and interfacing with the global search coordinator.

Two algorithms were used in the implementation described in [96] for both complete and partial solvers, the HGSADC of [169] and GUTS, a generalized version of the unified tabu search [31]. Briefly, HGSADC combines the exploration capability of population-based evolutionary search, the aggressive-improvement strength of neighborhood-based local search to enhance solutions newly created by genetic operators, and a solution evaluation function driven by both solution quality and contribution to the population diversity, which contributes to progress toward diverse and good solutions. GUTS is a tabu search-based meta-heuristic

implementing advanced insertion neighborhoods and allowing the exploration of unfeasible solutions by dynamically adjusting penalties on violations of vehicle capacity and route duration constraints. Both methods use relaxation of vehicle capacity and route duration constraints combined to penalization of infeasibilities in the evaluation function. They also use well-known VRP local neighborhoods based on pattern change, depot change, and inter- and intra-route movements.

Integrators build complete solutions by mixing partial solutions with promising features obtained within the PSGs. Integrators aim for solution quality, the transmission of critical features extracted from the partial solutions, and computational efficiency. Several Integrators can be involved in an ICS implementation, contributing to these goals and increasing the diversity of the population of complete solutions.

The simplest integrator consists in selecting high-quality partial solutions (with respect to solution value or the inclusion of particular decision combinations) and passing them directly to the complete solver group. Meta-heuristics, population-based methods in particular, e.g., genetic algorithms [169] and path relinking [131], may also be used, having proved their flexibility and stability in combining solution characteristics to yield high-quality solutions. Finally, a new methodology was proposed recently [65]. It proceeds through particular optimization models that preserve desired critical variables, defined as the variables whose values in the respective solution represent desired attributes, present in the partial solutions.

Four integrators were included in the MDPVRP application, the simple one passing good solutions to the CSG, and three others starting from pairs of partial solutions randomly selected among the best 25 % of the solutions in the central memories of the two PSGs. The second integrator applied the crossover operator of HGSADC and enhanced the new solution through the local search education operator of the same method. The third and fourth integrators applied the methodology of [65], the former aiming to transmit the attributes for which there was "consensus" in the input solutions, while the latter "promoted" them only through penalties added to the objective function.

The complete solver group (CSG) included a central memory, which included the complete solution set, as well as the context information and the guiding solutions built by the global search coordinator (GSC). The CSG received complete solutions from integrators and, when solvers were included (e.g., GUTS and HGSADC in the present case), enhanced them thus creating new ones. It was the global search coordinator which (1) built the search contextual information (e.g., the frequency of appearance of each (customer, depot, pattern) triplet in the complete solution set, together with the cost of the best solution containing it), (2) built new guiding solutions to orient the search toward promising features, and (3) monitored the status of the solver groups, sending guiding instructions (solutions) when necessary.

Monitoring is performed by following the evolution of the PSGs by, e.g., interrogating the central memories of the PSGs for the number of improving solutions generated during a certain time period. Monitoring provides the means to detect undesired situations, e.g., loss of diversity in the partial or complete populations, stagnation in improving the quality of the current best solution, awareness that some

zones of the solution space – defined by particular values for particular decision sets – have been scarcely explored, if at all, and that the search should be diversified in that direction, and so on. Whenever one of these criteria is not fulfilled, the GSC sends guidance "instructions" to the particular PSG. The particular type of guidance is application specific, but one may modify the values of the fixed attributes for the PSG to orient its search toward a different area or, more rarely, change the attribute subset under investigation (i.e., change the decomposition of the decision-set attributes) or modify/replace the solution method in a partial solver or integrator.

In the present case, the GSC guided the search trajectory of a particular PSG by sending three solutions, which were either randomly selected (equiprobably) from the complete solution set, or were three *guiding* solutions built by the GSC. The receiving PSG added directly these solutions to its own central memory, after resetting its population, all solutions being replaced by new randomly generated ones. Guiding solutions were continuously generated, and stored in a particular pool, to reflect the current status and the history of the search represented by the context information. The process proceeded by selecting promising triplets with respect to the search history, that is, triplets that appeared in at least one complete solution with a cost close (less than 3 % distant) to the current best solution. The promising triplets were used to create feasible pattern and depot customer assignments, routes being then generated by the local search of HGSADC to complete the solutions. These solutions were then individually enhanced by a short execution of GUTS or HGSADC.

Extensive experimental analyses were conducted to (1) assess the performance of ICS when compared to state-of-the-art sequential methods and (2) investigate a number of implementation alternatives. The general conclusions were that ICS performed extremely well. It obtained very good results even when compared to the state-of-the-art HGSADC meta-heuristic, obtaining several new best-known solutions in shorter computing times. The experiments also indicated that (1) one should use solvers displaying similar time performances in order to have all solvers contributing reasonably equally to the cooperation; (2) when using genetic solvers in a PSG, it is preferable for long runs to define a local population for each such solver and reserve the central memory of the PSG for communications and guidance only, while using the central memory as population for all cooperating genetic solvers is better for short runs; and (3) embedding good solvers (HGSADC in the present case) in the CSG enhances slightly the already excellent performance of the ICS parallel meta-heuristic.

Conclusions

This chapter presented an overview and state-of-the-art survey of the main parallel meta-heuristic ideas, discussing general concepts and algorithm design principles and strategies. The presentation was structured along the lines of a taxonomy of parallel meta-heuristics, which provided a rich framework for analyzing these

design principles and strategies, reviewing the literature, and identifying trends and promising research directions.

Four main classes of parallel meta-heuristics strategies may be identified: low-level decomposition of computing-intensive tasks with no modification to the original algorithm, decomposition of the search space, independent multi-search, and cooperative (multi) search, the later encompassing synchronous, asynchronous collegial, and knowledge-creating asynchronous collegial. It is noteworthy that this series also reflects the historical sequence of the development of parallel meta-heuristics. One should also note that, while the initial developments targeted genetic methods, simulated annealing, and tabu search, research is not addressing the full range of meta-heuristics. Furthermore, parallel meta-heuristics, cooperative search in particular, are now acknowledged as making up their own class of meta-heuristics.

Many important research questions and challenges exist for parallel meta-heuristics, in terms of general design methodology, instantiation to particular meta-heuristic frameworks and problem settings, and implementation on various computing architectures.

It is indeed noteworthy that despite the many years of research on these issues, there are still many gaps in knowledge, as well as in the studied meta-heuristic frameworks and problem classes. One may single out the many variants of swarm-based optimization and nongenetic population-based methods, scatter search and path relinking in particular. But one should not overlook the more classic meta-heuristic classes, as one still misses systematic and comprehensive/comparative studies of these issues. A large part of the studies present in the literature targeted combinatorial optimization problems with relatively few attributes and a single level of decision variables, e.g., vehicle routing problems. This is to be understood, these problems being important for science and practice and displaying large search spaces. Significant less research has been dedicated to multi-attribute problem settings, like the rich VRPs one increasingly has to tackle, and formulations with several levels of decisions like the single- and multilevel network design.

The community misses not only studies targeting particular meta-heuristic frameworks and problem classes but also transversal studies comparing the behavior and performance of particular parallel meta-heuristic strategies over different problem classes and of different parallel strategies and implementations for the same problem class. One increasingly finds such studies for sequential solution methods; we need them for parallel methods.

With respect to the four strategy classes, one should not forget that each fulfills a particular type of task and all are needed at some time. Thus, the idea that everything seems to be known regarding low-level parallelization strategies is not true. First, most studies on accelerating computing-intensive tasks targeted the evaluation of a population or neighborhood in classic meta-heuristic frameworks. These techniques should prove very valuable for swarm-based optimization, and more research is required in this field. Second, as shown in recent studies, the best strategy to accelerate a local search procedure may prove less effective when the local search is embedded into a full meta-heuristics or hierarchical solution methods. Third, the evolution of computing infrastructure opens up interesting but

challenging perspectives. Let's emphasize the possibilities offered by the graphic processing units, which increase continuously in power and are present everywhere, as surveyed in [16, 17].

Search-space decomposition also seems to have been thoroughly studied and has been overlooked in the last years, maybe due to the rapid and phenomenal increase in the memory available and the speed of access. Let's not forget, however, that most optimization problems of interest are complex and that the dimensions of the instances one faces in practice keep increasing. Research challenges exist in dynamic search-space decomposition and the combination of cooperative search and search-space decomposition. The integrative cooperative search is a first answer in this direction, but more research is needed.

Asynchronous cooperation, particularly when relaying on memories as inter-solver communication mechanisms, provides a powerful, flexible, and adaptable framework for parallel meta-heuristics that consistently achieved good results in terms of computing efficiency and solution quality for many meta-heuristic and problem classes. Other than the general research issues discussed above that are of particular interest in this context, a number of additional research issues and challenges are worth investigating.

A first issue concerns the exchange and utilization of context data locally generated by the cooperating solvers, to infer an image of the status of the global search and generate appropriate guiding instructions. Thus, contrasting the various local context data may be used to identify regions of the search space that were neglected or over explored. The information could also be used to evaluate the relative performance of the solvers conducting, eventually, to adjust the search parameters of particular solvers or even change the search strategy. So-called "strategic" decision variables or parameters could thus be more easily identified, which could prove very profitable in terms of search guidance.

A related issue concerns the learning processes and the creation of new information out of the shared data. Important questions concern the identification of information that may be derived from the exchanged solutions and context information and which is meaningful for, on the one hand, evaluating the status of the global search and, on the other hand, sending to solvers to guide their own search as part of the global optimization effort. Research in this direction is still at the very beginning but has already proved its worth, in particular in the context of the integrative cooperative methods.

A third broad issue concerns the cooperation of different types of meta-heuristics and of these exact solution methods. The so-called hybrid and matheuristic methods, representing the former and latter types of method combination, respectively, are trendy in the sequential optimization field. Very few studies explicitly target parallel methods. How different methods behave when involved in cooperative search and how the latter behaves given various combinations of methods is an important issue that should yield valuable insights into the design of parallel meta-heuristic algorithms, integrative cooperative search in particular. Actually, more research is required into ICS, both regarding its structure and components, and its application to various problem settings. A particularly challenging but fascinating direction for

cooperative search and ICS is represented by the multi-scenario representation of stochastic optimization formulations, for which almost nothing beyond low-level scenario decomposition has been proposed.

Finally, the issue of understanding cooperation on some fundamental level, giving the means to formally define and analyze it in order to design better, more efficient algorithms. As mentioned earlier, this work parallels efforts in many other scientific domains addressing issues related to emerging decision and behavior out of the decisions and behaviors or several independent entities. Theoretical and empirical work is needed in order to address this fascinating and difficult question.

Acknowledgments The author wishes to acknowledge the contributions of colleagues and students, in particular Professors Michel Gendreau, Université de Montréal, Canada, and Michel Toulouse, the Vietnamese-German University, Vietnam, who collaborated over the years to the work on parallel meta-heuristics for combinatorial optimization. All errors are solely and entirely due to the author, however.

Partial funding for this project has been provided by the Natural Sciences and Engineering Council of Canada (NSERC), through its Discovery Grant and the Discovery Accelerator Supplements programs, and the Strategic Clusters program of the Fonds québécois de la recherche sur la nature et les technologies. The author thanks the two institutions for supporting this research.

References

1. Aiex RM, Martins SL, Ribeiro CC, Rodriguez NR (1998) Cooperative multi-thread parallel tabu search with an application to circuit partitioning. In: Proceedings of IRREGULAR'98 – 5th international symposium on solving irregularly structured problems in parallel. Lecture notes in computer science, vol 1457. Springer, Berlin/New York, pp 310–331
2. Alba E (ed) (2005) Parallel metaheuristics: a new class of algorithms. Wiley, Hoboken
3. Alba E, Dorronsoro B (2004) Solving the vehicle routing problem by using cellular genetic algorithms. In: Gottlieb J, Günther RR (eds) Evolutionary computation in combinatorial optimization, 4th European conference, EvoCOP 2004, Coimbra, 5–7 Apr 2004. Lecture notes in computer science, vol 3004. Springer, Heidelberg, pp 11–20
4. Alba E, Luque G, Nesmachnow S (2013) Parallel metaheuristics: recent advances and new trends. Int Trans Oper Res 20(1):1–48
5. Azencott R (1992) Simulated annealing parallelization techniques. Wiley, New York
6. Badeau P, Gendreau M, Guertin F, Potvin JY, Taillard E (1997) A parallel tabu search heuristic for the vehicle routing problem with time windows. Transp Res C: Emerg Technol 5(2): 109–122
7. Banos R, Gil C, Ortega J, Montoya FG (2004) A parallel multilevel metaheuristic for graph partitioning. J Heuristics 10(4):315–336
8. Banos R, Gil C, Ortega J, Montoya FG (2004) Parallel heuristic search in multilevel graph partitioning. In: Proceedings of the 12th Euromicro conference on parallel, distributed and network-based processing, A Coruña, pp 88–95
9. Barr RS, Hickman BL (1993) Reporting computational experiments with parallel algorithms: issues, measures, and experts opinions. ORSA J Comput 5(1):2–18
10. Bastos MP, Ribeiro CC (1999) Reactive tabu search with path-relinking for the Steiner problem in graphs. In: Voß S, Martello S, Roucairol C, Osman IH (eds) Meta-heuristics 98: theory & applications. Kluwer Academic, Norwell, pp 31–36
11. Battiti R, Tecchiolli G (1992) Parallel based search for combinatorial optimization: genetic algorithms and TABU. Microprocessors Microsyst 16(7):351–367

12. Berger J, Barkaoui M (2004) A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. Comput Oper Res 31(12):2037–2053
13. Blazewicz J, Moret-Salvador A, Walkowiak R (2004) Parallel tabu search approaches for two-dimensional cutting. Parallel Process Lett 14(1):23–32
14. Bock S, Rosenberg O (2000) A new parallel breadth first tabu search technique for solving production planning problems. Int Trans Oper Res 7(6):625–635
15. Bortfeldt A, Gehring H, Mack D (2003) A parallel tabu search algorithm for solving the container loading problem. Parallel Comput 29:641–662
16. Brodtkorb AR, Hagen TR, Schulz C, Hasle G (2013) GPU computing in discrete optimization. Part I: introduction to the GPU. EURO J Transp Logist 2(1–2):129–157
17. Brodtkorb AR, Hagen TR, Schulz C, Hasle G (2013) GPU computing in discrete optimization. Part II: survey focussed on routing problems. EURO J Transp Logist 2(1–2):159–186
18. Bullnheimer B, Kotsis G, Strauß C (1999) Parallelization strategies for the ant system. In: De Leone R, Murli A, Pardalos P, Toraldo G (eds) High performance algorithms and software in nonlinear optimization. Applied optimization, vol 24, Kluwer Academic, Dordrecht, pp 87–100. http://www.bwl.univie.ac.at/bwl/prod/papers/pom-wp-9-97.ps
19. Calégari P, Guidec F, Kuonen P, Kuonen D (1997) Parallel Island-based genetic algorithm for radio network design. J Parallel Distrib Comput 47(1):86–90
20. Cantú-Paz E (1998) A survey of parallel genetic algorithms. Calculateurs Parallèles, Réseaux et Systèmes répartis 10(2):141–170
21. Cantú-Paz E (2005) Theory of parallel genetic algorithms. In: Alba E (ed) Parralel metaheuristics: a new class of algorithms. Wiley, Hoboken, pp 425–445
22. Cavalcante CBC, Cavalcante VF, Ribeiro CC, Souza MC (2002) Parallel cooperative approaches for the labor constrained scheduling problem. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer Academic, Norwell, pp 201–225
23. Chakrapani J, Skorin-Kapov J (1992) A connectionist approach to the quadratic assignment problem. Comput Oper Res 19(3/4):287–295
24. Chakrapani J, Skorin-Kapov J (1993) Connection machine implementation of a tabu search algorithm for the traveling salesman problem. J Comput Inf Technol 1(1):29–36
25. Chakrapani J, Skorin-Kapov J (1993) Massively parallel tabu search for the quadratic assignment problem. Ann Oper Res 41:327–341
26. Chao IM, Golden B L, Wasil EA (1995) An improved heuristic for the period vehicle routing problem. Networks 26(1):25–44
27. Cohoon J, Hedge S, Martin W, Richards D (1987) Punctuated equilibria: a parallel genetic algorithm. In: Grefenstette J (ed) Proceedings of the second international conference on genetic algorithms and their applications. Lawrence Erlbaum Associates, Hillsdale, pp 148–154
28. Cohoon J, Hedge S, Richards D (1991) Genetic algorithm and punctuated equilibria in VLSI. In: Schwefel H-P, Männer R (eds) Parallel problem solving from nature. Lecture notes in computer science, vol 496. Springer, Berlin, pp 134–144
29. Cohoon J, Hedge S, Richards D (1991) A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes. In: Belew R, Booker L (eds) Proceedings of the fourth international conference on genetic algorithms. Morgan Kaufmann, San Mateo, pp 134–144
30. Cordeau JF, Maischberger M (2012) A parallel iterated tabu search heuristic for vehicle routing problems. Comput Oper Res 39(9):2033–2050
31. Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. J Oper Res Soc 52:928–936
32. Crainic TG (2005) Parallel computation, co-operation, tabu search. In: Rego C, Alidaee B (eds) Metaheuristic optimization via memory and evolution: tabu search and scatter search. Kluwer Academic, Norwell, pp 283–302
33. Crainic TG (2008) Parallel solution methods for vehicle routing problems. In: Golden BL, Raghavan S, Wasil EA (eds) The vehicle routing problem: latest advances and new challenges. Springer, New York, pp 171–198

34. Crainic TG, Gendreau M (1999) Towards an evolutionary method – cooperating multi-thread parallel tabu search hybrid. In: Voß S, Martello S, Roucairol C, Osman IH (eds) Meta-heuristics 98: theory & applications. Kluwer Academic, Norwell, pp 331–344
35. Crainic TG, Gendreau M (2002) Cooperative parallel tabu search for capacitated network design. J Heuristics 8(6):601–627
36. Crainic TG, Hail N (2005) Parallel meta-heuristics applications. In: Alba E (ed) Parallel metaheuristics: a new class of algorithms. Wiley, Hoboken, pp 447–494
37. Crainic TG, Toulouse M (1998) Parallel metaheuristics. In: Crainic TG, Laporte G (eds) Fleet management and logistics. Kluwer Academic, Norwell, pp 205–251
38. Crainic TG, Toulouse M (2003) Parallel strategies for meta-heuristics. In: Glover F, Kochenberger G (eds) Handbook in metaheuristics. Kluwer Academic, Norwell, pp 475–513
39. Crainic TG, Toulouse M (2008) Explicit and emergent cooperation schemes for search algorithms. In: Maniezzo V, Battiti R, Watson J-P (eds) Learning and intelligent optimization. Lecture notes in computer science, vol 5315. Springer, Berlin, pp 95–109
40. Crainic TG, Toulouse M (2010) Parallel meta-heuristics. In: Gendreau M, Potvin J-Y (eds) Handbook of metaheuristics, 2nd edn. Springer, New York, pp 497–541
41. Crainic TG, Toulouse M, Gendreau M (1995) Synchronous tabu search parallelization strategies for multicommodity location-allocation with balancing requirements. OR Spektrum 17(2/3):113–123
42. Crainic TG, Toulouse M, Gendreau M (1996) Parallel asynchronous tabu search for multicommodity location-allocation with balancing requirements. Ann Oper Res 63:277–299
43. Crainic TG, Toulouse M, Gendreau M (1997) Towards a taxonomy of parallel tabu search algorithms. INFORMS J Comput 9(1):61–72
44. Crainic TG, Gendreau M, Hansen P, Mladenović N (2004) Cooperative parallel variable neighborhood search for the p-median. J Heuristics 10(3):293–314
45. Crainic TG, Gendreau M, Potvin JY (2005) Parallel tabu search. In: Alba E (ed) Parallel metaheuristics. Wiley, Hoboken, pp 298–313
46. Crainic TG, Di Chiara B, Nonato M, Tarricone L (2006) Tackling electrosmog in completely configured 3G networks by parallel cooperative meta-heuristics. IEEE Wirel Commun 13(6):34–41
47. Crainic TG, Li Y, Toulouse M (2006) A first multilevel cooperative algorithm for the capacitated multicommodity network design. Comput Oper Res 33(9):2602–2622
48. Crainic TG, Crisan GC, Gendreau M, Lahrichi N, Rei W (2009) A concurrent evolutionary approach for cooperative rich combinatorial optimization. In: Genetic and evolutionary computation conference – GECCO 2009, Montréal, 8–12 July. ACM, cD-ROM
49. Crainic TG, Crisan GC, Gendreau M, Lahrichi N, Rei W (2009) Multi-thread integrative cooperative optimization for rich combinatorial problems. In: The 12th international workshop on nature inspired distributed computing – NIDISC'09, 25–29 May, Rome, cD-ROM
50. Crainic TG, Davidović T, Ramljak D (2014) Designing parallel meta-heuristic methods. In: Despotovic-Zrakic M, Milutinovic V, Belic A (eds) High performance and cloud computing in scientific research and education. IGI Global, Hershey, pp 260–280
51. Cung VD, Martins SL, Ribeiro CC, Roucairol C (2002) Strategies for the parallel implementations of metaheuristics. In: Ribeiro C, Hansen P (eds) Essays and surveys in metaheuristics. Kluwer Academic, Norwell, pp 263–308
52. Czech ZJ (2000) A parallel genetic algorithm for the set partitioning problem. In: 8th Euromicro workshop on parallel and distributed processing, Rhodos, pp 343–350
53. Dai C, Li B, Toulouse M (2009) A multilevel cooperative tabu search algorithm for the covering design problem. J Comb Math Comb Comput 68:35–65
54. Davidović T, Crainic TG (2015) Parallel local search to schedule communicating tasks on identical processors. Parallel Comput 48:1–14
55. De Falco I, Del Balio R, Tarantino E, Vaccaro R (1994) Improving search by incorporating evolution principles in parallel tabu search. In: Proceedings international conference on machine learning, New Brunswick, pp 823–828

56. De Falco I, Del Balio R, Tarantino E (1995) Solving the mapping problem by parallel tabu search. Report, Istituto per la Ricerca sui Sistemi Informatici Paralleli-CNR
57. Di Chiara B (2006) Optimum planning of 3G cellular systems: radio propagation models and cooperative parallel meta-heuristics. PhD thesis, Dipartimento di ingegneria dell'innovatione, Universitá degli Studi di Lecce, Lecce
58. Diekmann R, Lüling R, Monien B, Spräner C (1996) Combining helpful sets and parallel simulated annealing for the graph-partitioning problem. Int J Parallel Program 8:61–84
59. Doerner K, Hartl RF, Kiechle G, Lucka M, Reimann M (2004) Parallel ant systems for the capacitated vehicle routing problem. In: Gottlieb J, Raidl GR (eds) Evolutionary computation in combinatorial optimization: 4th European conference, EvoCOP 2004. Lecture notes in computer science, vol 3004. Springer, Berlin, pp 72–83
60. Doerner KF, Hartl RF, Lucka M (2005) A parallel version of the D-ant algorithm for the vehicle routing problem. In: Vajtersic M, Trobec R, Zinterhof P, Uhl A (eds) Parallel numerics'05. Springer, New York, pp 109–118
61. Doerner KF, Hartl RF, Benkner S, Lucka M (2006) Cooperative savings based ant colony optimization – multiple search and decomposition approaches. Parallel Process Lett 16(3):351–369
62. Dorigo M, Stuetzle T (2003) The ant colony metaheuristic. Algorithms, applications, and advances. In: F Glover, G Kochenberger (eds) Handbook in metaheuristics. Kluwer Academic, Norwell, pp 251–285
63. Dorronsoro B, Arias F, Luna A, Nebro AJ, Alba E (2007) A grid-based hybrid cellular genetic algorithm for very large scale instances of the CVRP. In: Smari W (ed) High performance computing & simulation conference HPCS 2007 within the 21st European conference on modelling and simulation ECMS 2007, pp 759–765. http://www.scs-europe.net/conf/ecms2007/ecms2007-cd/ecms2007/ecms2007finalpapers.html
64. Drias H, Ibri A (2003) Parallel ACS for weighted MAX-SAT. In: Mira J, Álvarez J (eds) Artificial neural nets problem solving methods – proceedings of the 7th international work-conference on artificial and natural neural networks. Lecture notes in computer science, vol 2686. Springer, Heidelberg, pp 414–421
65. El Hachemi N, Crainic TG, Lahrichi N, Rei W, Vidal T (2014) Solution integration in combinatorial optimization with applications to cooperative search and rich vehicle routing. Publication CIRRELT-2014-40, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport, Université de Montréal, Montréal
66. Fiechter CN (1994) A parallel tabu search algorithm for large travelling salesman problems. Discrete Appl Math 51(3):243–267
67. Flores CD, Cegla BB, Caceres DB (2003) Telecommunication network design with parallel multi-objective evolutionary algorithms. In: IFIP/ACM Latin America networking conference 2003, La Paz
68. Folino G, Pizzuti C, Spezzano G (1998) Combining cellular genetic algorithms and local search for solving satisfiability problems. In: Proceedings of the tenth IEEE international conference on tools with artificial intelligence. IEEE Computer Society Press, Piscataway, pp 192–198
69. Folino G, Pizzuti C, Spezzano G (1998) Solving the satisfiability problem by a parallel cellular genetic algorithm. In: Proceedings of the 24th EUROMICRO conference. IEEE Computer Society Press, Los Alamitos, pp 715–722
70. Garcia BL, Potvin JY, Rousseau JM (1994) A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. Comput Oper Res 21(9):1025–1033
71. García-López F, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2002) The parallel variable neighborhood search for the p-median problem. J Heuristics 8(3):375–388
72. García-López F, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2003) Parallelization of the scatter search for the p-median problem. Parallel Comput 29:575–589
73. García-López F, García Torres M, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2005) Parallel scatter search. In: Alba E (ed) Parallel metaheuristics: a new class of metaheuristics. Wiley, Hoboken, pp 223–246

74. García-López F, García Torres M, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM (2006) Solving feature subset selection problem by a parallel scatter search. Eur J Oper Res 169:477–489
75. Gehring H, Homberger J (1997) A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, M akel]:a M, Toivanen J (eds) Proceedings of EUROGEN99 – short course on evolutionary algorithms in engineering and computer science, Jyvskyla, pp 57–64
76. Gehring H, Homberger J (2001) A parallel two-phase metaheuristic for routing problems with time windows. Asia-Pac J Oper Res 18(1):35–47
77. Gehring H, Homberger J (2002) Parallelization of a two-phase metaheuristic for routing problems with time windows. J Heuristics 8:251–276
78. Gendreau M, Hertz A, Laporte G (1994) A tabu search heuristic for the vehicle routing problem. Manag Sci 40:1276–1290
79. Gendreau M, Guertin F, Potvin JY, Taillard ÉD (1999) Tabu search for real-time vehicle routing and dispatching. Transp Sci 33(4):381–390
80. Gendreau M, Laporte G, Semet F (2001) A dynamic model and parallel tabu search heuristic for real-time ambulance relocation. Parallel Comput 27(12):1641–1653
81. Glover F (1996) Tabu search and adaptive memory programming – advances, applications and challenges. In: Barr R, Helgason R, Kennington J (eds) Interfaces in computer science and operations research. Kluwer Academic, Norwell, pp 1–75
82. Glover F, Laguna M (1997) Tabu search. Kluwer Academic, Norwell,
83. Golden B L, Wasil EA, Kelly JP, Chao IM (1998) Metaheuristics in vehicle routing. In: Crainic T, Laporte G (eds) Fleet management and logistics. Kluwer Academic, Norwell, pp 33–56
84. Greening DR (1989) A taxonomy of parallel simulated annealing techniques. Technical report No. RC 14884, IBM
85. Greening DR (1990) Asynchronous parallel simulated annealing. Lect Complex Syst 3:497–505
86. Greening DR (1990) Parallel simulated annealing techniques. Physica D 42:293–306
87. Groër C, Golden B (2011) A parallel algorithm for the vehicle routing problem. INFORMS J Comput 23(2):315–330
88. Herdy M (1992) Reproductive isolation as strategy parameter in hierarchical organized evolution strategies. In: Männer R, Manderick B (eds) Parallel problem solving from nature, 2. North-Holland, Amsterdam, pp 207–217
89. Hidalgo JI, Prieto M, Lanchares J, Baraglia R, Tirado F, Garnica O (2003) Hybrid parallelization of a compact genetic algorithm. In: Proceedings of the 11th Euromicro conference on parallel, distributed and network-based processing, Genova, pp 449–455
90. Holmqvist K, Migdalas A, Pardalos PM (1997) Parallelized heuristics for combinatorial search. In: Migdalas A, Pardalos P, Storoy S (eds) Parallel computing in optimization. Kluwer Academic, Norwell, pp 269–294
91. Homberger J, Gehring H (1999) Two evolutionary metaheuristics for the vehicle routing problem with time windows. INFOR 37:297–318
92. Janson S, Merkle D, Middendorf M (2005) Parallel ant colony algorithms. In: Alba E (ed) Parallel metaheuristics: a new class of metaheuristics. Wiley, Hoboken, pp 171–201
93. Jin J, Crainic TG, Løkketangen A (2012) A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. Eur J Oper Res 222(3):441–451
94. Jin J, Crainic TG, Løkketangen A (2014) A cooperative parallel metaheuristic for the capacitated vehicle routing problems. Comput Oper Res 44:33–41
95. Laganière R, Mitiche A (1995) Parallel tabu search for robust image filtering. In: Proceedings of IEEE workshop on nonlinear signal and image processing (NSIP'95), Neos Marmaras, vol 2, pp 603–605
96. Lahrichi N, Crainic TG, Gendreau M, Rei W, Crisan CC, Vidal T (2015) An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. Eur J Oper Res 246(2):400–412

97. Laursen PS (1996) Parallel heuristic search – introductions and a new approach. In: Ferreira A, Pardalos P (eds) Solving combinatorial optimization problems in parallel. Lecture notes in computer science, vol 1054. Springer, Berlin, pp 248–274
98. Le Bouthillier A (2007) Recherches coopératives pour la résolution de problèmes d'optimisation combinatoire. PhD thesis, Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal
99. Le Bouthillier A, Crainic TG (2005) A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Comput Oper Res 32(7):1685–1708
100. Le Bouthillier A, Crainic TG, Kropf P (2005) A guided cooperative search for the vehicle routing problem with time windows. IEEE Intell Syst 20(4):36–42
101. Lee KG, Lee SY (1992) Efficient parallelization of simulated annealing using multiple Markov chains: an application to graph partitioning. In: Mudge TN (ed) Proceedings of the international conference on parallel processing. Algorithms and applications, vol III. CRC Press, Boca Raton, pp 177–180
102. Lee SY, Lee KG (1992) Asynchronous communication of multiple Markov chains in parallel simulated annealing. In: Mudge TN (ed) Proceedings of the international conference on parallel processing. Algorithms and applications, vol III. CRC Press, Boca Raton, pp 169–176
103. Lee KG, Lee SY (1995) Synchronous and asynchronous parallel simulated annealing with multiple Markov chains. In: Brandenburg F (ed) Graph drawing – proceedings GD '95, symposium on graph drawing, Passau. Lecture notes in computer science, vol 1027. Springer, Berlin, pp 396–408
104. Lee SY, Lee KG (1996) Synchronous and asynchronous parallel simulated annealing with multiple Markov chains. IEEE Trans Parallel Distrib Syst 7(10):993–1007
105. Li Y, Pardalos PM, Resende MGC (1994) A greedy randomized adaptive search procedure for quadratic assignment problem. In: DIMACS implementation challenge. DIMACS series on discrete mathematics and theoretical computer science, vol 16. American Mathematical Society, pp 237–261
106. Li F, Golden B L, Wasil EA (2005) A very large-scale vehicle routing: new test problems, algorithms, and results. Comput Oper Res 32(5):1165–1179
107. Lin SC, Punch WF, Goodman ED (1994) Coarse-grain parallel genetic algorithms: categorization and new approach. In: Sixth IEEE symposium on parallel and distributed processing. IEEE Computer Society Press, Los Alamitos, pp 28–37
108. Luque G, Alba E, Dorronsoro B (2005) Parallel genetic algorithms. In: Alba E (ed) Parralel metaheuristics: a new class of algorithms. Wiley, Hoboken, pp 107–125
109. Malek M, Guruswamy M, Pandya M, Owens H (1989) Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Ann Oper Res 21:59–84
110. Martins SL, Ribeiro CC, Souza MC (1998) A parallel GRASP for the Steiner problem in graphs. In: Ferreira A, Rolim J (eds) Proceedings of IRREGULAR'98 – 5th international symposium on solving irregularly structured problems in parallel. Lecture notes in computer science, vol 1457. Springer, Berlin/New York, pp 285–297
111. Martins SL, Resende MGC, Ribeiro CC, Parlados PM (2000) A parallel grasp for the Steiner tree problem in graphs using a hybrid local search strategy. J Glob Optim 17:267–283
112. Michels R, Middendorf M (1999) An ant system for the shortest common supersequence problem. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, London, pp 51–61
113. Middendorf M, Reischle F, Schmeck H (2002) Multi colony ant algorithms. J Heuristics 8(3):305–320. doihttp://dx.doi.org/10.1023/A:1015057701750
114. Miki M, Hiroyasu T, Wako J, Yoshida T (2003) Adaptive temperature schedule determined by genetic algorithm for parallel simulated annealing. In: CEC'03 – the 2003 congress on evolutionary computation, Canberra, vol 1, pp 459–466
115. Mingozzi A (2005) The multi-depot periodic vehicle routing problem. In: Abstraction, reformulation and approximation. Lecture notes in computer science. Springer, Berlin/Heidelberg, pp 347–350

116. Moreno-Pérez JA, Hansen P, Mladenović N (2005) Parallel variable neighborhood search. In: Alba E (ed) Parallel metaheuristics: a new class of metaheuristics. Wiley, Hoboken, pp 247–266
117. Mühlenbein H (1989) Parallel genetic algorithms, population genetics and combinatorial optimization. In: Schaffer J (ed) Proceedings of the third international conference on genetic algorithms. Morgan Kaufmann, San Mateo, pp 416–421
118. Mühlenbein H (1991) Parallel genetic algorithms, population genetics, and combinatorial optimization. In: Becker JD, Eisele I, Mündemann FW (eds) Parallelism, learning, evolution. Workshop on evolutionary models and strategies – WOPPLOT 89. Springer, Berlin, pp 398–406
119. Mühlenbein H (1992) Parallel genetic algorithms in combinatorial optimization. In: Balci O, Sharda R, Zenios S (eds) Computer science and operations research: new developments in their interface. Pergamon Press, New York, pp 441–456
120. Niar S, Fréville A (1997) A parallel tabu search algorithm for the 0–1 multidimensional knapsack problem. In: 11th international parallel processing symposium (IPPS '97), Geneva. IEEE, pp 512–516
121. Oduntan I, Toulouse M, Baumgartner R, Bowman C, Somorjai R, Crainic TG (2008) A multilevel tabu search algorithm for the feature selection problem in biomedical data sets. Comput Math Appl 55(5):1019–1033
122. Ouyang M, Toulouse M, Thulasiraman K, Glover F, Deogun JS (2000) Multi-level cooperative search: application to the netlist/hypergraph partitioning problem. In: Proceedings of international symposium on physical design. ACM, New York, pp 192–198
123. Ouyang M, Toulouse M, Thulasiraman K, Glover F, Deogun JS (2002) Multilevel cooperative search for the circuit/hypergraph partitioning problem. IEEE Trans Comput-Aided Des 21(6):685–693
124. Pardalos PM, Li Y, KA M (1992) Computational experience with parallel algorithms for solving the quadratic assignment problem. In: Balci O, Sharda R, Zenios S (eds) Computer science and operations research: new developments in their interface. Pergamon Press, New York, pp 267–278
125. Pardalos PM, Pitsoulis L, Mavridou T, Resende MGC (1995) Parallel search for combinatorial optimization: genetic algorithms, simulated annealing, tabu search and GRASP. In: Ferreira A, Rolim J (eds) Proceedings of workshop on parallel algorithms for irregularly structured problems. Lecture notes in computer science, vol 980. Springer, Berlin, pp 317–331
126. Pardalos PM, Pitsoulis L, Resende MGC (1995) A parallel GRASP implementation for the quadratic assignment problem. In: Ferreira A, Rolim J (eds) Solving irregular problems in parallel: state of the art. Kluwer Academic, Norwell, pp 115–130
127. Polacek M, Benkner S, Doerner KF, Hartl RF (2008) A cooperative and adaptive variable neighborhood search for the multi depot vehicle routing problem with time windows. Bus Res 1(2):1–12
128. Porto SCS, Ribeiro CC (1995) A tabu search approach to task scheduling on heteregenous processors under precedence constraints. Int J High-Speed Comput 7:45–71
129. Porto SCS, Ribeiro CC (1996) Parallel tabu search message-passing synchronous strategies for task scheduling under precedence constraints. J Heuristics 1(2):207–223
130. Porto SCS, Kitajima JPFW, Ribeiro CC (2000) Performance evaluation of a parallel tabu search task scheduling algorithm. Parallel Comput 26:73–90
131. Rahimi Vahed A, Crainic TG, Gendreau M, Rei W (2013) A path relinking algorithm for a multi-depot periodic vehicle routing problem. J Heuristics 19(3):497–524
132. Rahoual M, Hadji R, Bachelet V (2002) Parallel ant system for the set covering problem. In: Dorigo M, Di Caro G, Sampels M (eds) Ant algorithms – proceedings of the third international workshop, ANTS 2002. Lecture notes in computer science, vol 2463. Springer, Berlin, pp 262–267
133. Ram DJ, Sreenivas TH, Subramaniam KG (1996) Parallel simulated annealing algorithms. J Parallel Distrib Comput 37:207–212

134. Randall M, Lewis A (2002) A parallel implementation of ant colony optimisation. J Parallel Distrib Comput 62:1421–1432
135. Rego C (2001) Node ejection chains for the vehicle routing problem: sequential and parallel algorithms. Parallel Comput 27:201–222
136. Rego C, Roucairol C (1996) A parallel tabu search algorithm using ejection chains for the VRP. In: Osman I, Kelly J (eds) Meta-heuristics: theory & applications. Kluwer Academic, Norwell, pp 253–295
137. Reimann M, Stummer M, Doerner K (2002) A savings based ants system for the vehicle routing problem. In: Langton C, Cantú-Paz E, Mathias KE, Roy R, Davis L, Poli R, Balakrishnan K, Honavar V, Rudolph G, Wegener J, Bull L, Potter MA, Schultz AC, Miller JF, Burke EK, Jonoska N (eds) GECCO 2002: proceedings of the genetic and evolutionary computation conference, New York, 9–13 July 2002. Morgan Kaufmann, San Francisco, pp 1317–1326
138. Reimann M, Doerner K, Hartl R (2004) D-ants: savings based ants divide and conquer the vehicle routing problem. Comput Oper Res 31(4):563–591
139. Ribeiro CC, Rosseti I (2002) A parallel GRASP heuristic for the 2-path network design problem. 4 journée ROADEF, Paris, 20–22 Feb
140. Ribeiro CC, Rosseti I (2002) A parallel GRASP heuristic for the 2-path network design problem. Third meeting of the PAREO Euro working group, Guadeloupe, May
141. Ribeiro CC, Rosseti I (2002) Parallel grasp with path-relinking heuristic for the 2-path network design problem. In: AIRO'2002, L'Aquila, Sept
142. Rochat Y, Taillard ED (1995) Probabilistic diversification and intensification in local search for vehicle routing. J Heuristics 1(1):147–167
143. Sanvicente-Sánchez H, Frausto-Solís J (2002) MPSA: a methodology to parallelize simulated annealing and its application to the traveling salesman problem. In: Coello Coello C, de Albornoz A, Sucar L, Battistutti O (eds) MICAI 2002: advances in artificial intelligence. Lecture notes in computer science, vol 2313. Springer, Heidelberg, pp 89–97
144. Schlierkamp-Voosen D, Mühlenbein H (1994) Strategy adaptation by competing subpopulations. In: Davidor Y, Schwefel H-P, Männer R (eds) Parallel problem solving from nature III. Lecture notes in computer science, vol 866. Springer, Berlin, pp 199–208
145. Schulze J, Fahle T (1999) A parallel algorithm for the vehicle routing problem with time window constraints. Ann Oper Res 86:585–607
146. Sevkli M, Aydin ME (2007) Parallel variable neighbourhood search algorithms for job shop scheduling problems. IMA J Manag Math 18(2):117–133
147. Shonkwiler R (1993) Parallel genetic algorithms. In: Forrest S (ed) Proceedings of the fifth international conference on genetic algorithms. Morgan Kaufmann, San Mateo, pp 199–205
148. Solar M, Parada V, Urrutia R (2002) A parallel genetic algorithm to solve the set-covering problem. Comput Oper Res 29(9):1221–1235
149. Stutzle T (1998) Parallelization strategies for ant colony optimization. In: Eiben AE, Back T, Schoenauer M, Schwefel H-P (eds) Proceedings of parallel problem solving from nature V. Lecture notes in computer science, vol 1498. Springer, Heidelberg, pp 722–731
150. Taillard ÉD (1991) Robust taboo search for the quadratic assignment problem. Parallel Comput 17:443–455
151. Taillard ÉD (1993) Parallel iterative search methods for vehicle routing problems. Networks 23:661–673
152. Taillard ÉD (1994) Parallel taboo search techniques for the job shop scheduling problem. ORSA J Comput 6(2):108–117
153. Taillard ÉD, Gambardella LM, Gendreau M, Potvin JY (1997) Adaptive memory programming: a unified view of metaheuristics. Eur J Oper Res 135:1–10
154. Taillard ÉD, Gambardella LM, Gendreau M, Potvin JY (1998) Programmation à mémoire adaptative. Calculateurs Parallèles, Réseaux et Systèmes répartis 10:117–140
155. Talbi EG (ed) (2006) Parallel combinatorial optimization. Wiley, Hoboken

156. Talbi EG, Hafidi Z, Geib JM (1998) Parallel adaptive tabu search approach. Parallel Comput 24:2003–2019
157. Talbi EG, Roux O, Fonlupt C, Robillard D (1999) Parallel ant colonies for combinatorial optimization problems. In: Rolim J et al (ed) 11th IPPS/SPDP'99 workshops held in conjunction with the 13th international parallel processing symposium and 10th symposium on parallel and distributed processing, San Juan, 12–16 Apr. Lecture notes in computer science, vol 1586. Springer, Berlin, pp 239–247
158. Talukdar S, Baerentzen L, Gove A, de Souza P (1998) Asynchronous teams: cooperation schemes for autonomous agents. J Heuristics 4:295–321
159. Talukdar S, Murthy S, Akkiraju R (2003) Assynchronous teams. In: Glover F, Kochenberger G (eds) Handbook in metaheuristics. Kluwer Academic, Norwell, pp 537–556
160. ten Eikelder HMM, Aarts BJL, Verhoeven MGA, Aarts EHL (1999) Sequential and parallel local search for job shop scheduling. In: Voß S, Martello S, Roucairol C, Osman IH (eds) Meta-heuristics 98: theory & applications. Kluwer Academic, Norwell, pp 359–371
161. Tongcheng G, Chundi M (2002) Radio network design using coarse-grained parallel genetic algorithms with different neighbor topology. In: Proceedings of the 4th world congress on intelligent control and automation, Shanghai, vol 3, pp 1840–1843
162. Toulouse M, Crainic TG, Gendreau M (1996) Communication issues in designing cooperative multi thread parallel searches. In: Osman IH, Kelly JP (eds) Meta-heuristics: theory & applications. Kluwer Academic, Norwell, pp 501–522
163. Toulouse M, Crainic TG, Sansó B, Thulasiraman K (1998) Self-organization in cooperative search algorithms. In: Proceedings of the 1998 IEEE international conference on systems, man, and cybernetics. Omnipress, Madisson, pp 2379–2385
164. Toulouse M, Crainic TG, Sansó B (1999) An experimental study of systemic behavior of cooperative search algorithms. In: Voß S, Martello S, Roucairol C, Osman IH (eds) Meta-heuristics 98: theory & applications. Kluwer Academic, Norwell, pp 373–392
165. Toulouse M, Thulasiraman K, Glover F (1999) Multi-level cooperative search: a new paradigm for combinatorial optimization and an application to graph partitioning. In: Amestoy P, Berger P, Daydé M, Duff I, Frayssé V, Giraud L, Ruiz D (eds) 5th international Euro-par parallel processing conference. Lecture notes in computer science, vol 1685. Springer, Heidelberg, pp 533–542
166. Toulouse M, Crainic TG, Thulasiraman K (2000) Global optimization properties of parallel cooperative search algorithms: a simulation study. Parallel Comput 26(1):91–112
167. Toulouse M, Crainic TG, Sansó B (2004) Systemic behavior of cooperative search algorithms. Parallel Comput 30(1):57–79
168. Verhoeven MGA, Aarts EHL (1995) Parallel local search. J Heuristics 1(1):43–65
169. Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multi-depot and periodic vehicle routing problems. Oper Res 60(3):611–624
170. Voß S (1993) Tabu search: applications and prospects. In: Du DZ, Pardalos P (eds) Network optimization problems. World Scientific, Singapore, pp 333–353
171. Wilkerson R, Nemer-Preece N (1998) Parallel genetic algorithm to solve the satisfiability problem. In: Proceedings of the 1998 ACM symposium on applied computing. ACM, New York, pp 23–28